NASA/CR—2006-214238

Propulsion IVHM Technology Experiment

Amy K. Chicatelli, William A. Maul, and Christopher E. Fulton
Analex Corporation, Brook Park, Ohio

July 2006

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 301–621–0134

- Telephone the NASA STI Help Desk at 301–621–0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076–1320

NASA/CR—2006-214238

Propulsion IVHM Technology Experiment

Amy K. Chicatelli, William A. Maul, and Christopher E. Fulton
Analex Corporation, Brook Park, Ohio

Prepared under Contract NAS3–00145

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

July 2006

Acknowledgments

The authors would like to acknowledge the contributions of all of the other PITEX and NITEX team members at the NASA Ames Research Center, Glenn Research Center, and Kennedy Space Center. Their contributions were instrumental in the successful completion of this work. The authors would also like to acknowledge the past support and guidance of the NGLT IVHM Project Office management by Ames Research Center.

Level of Review: This material has been technically reviewed by an expert single reviewer.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Available electronically at http://gltrs.grc.nasa.gov

Contents

List of Figures

List of Tables

1.0 Introduction

The Propulsion IVHM (Integrated Vehicle Health Management) Technology Experiment (PITEX) was a NASA Glenn Research Center (GRC) led sub-task under Northrop Grumman Corporation's (NGC) contract for the Space Launch Initiative (SLI) Program. The PITEX objective was to mature and demonstrate key IVHM technologies on a relevant 2nd Generation Reusable Launch Vehicle (2GRLV) propulsion system in support of NASA's goal to improve the safety, affordability, and reliability of future space transportation systems. The PITEX research was based on legacy work, funded through the Future-X Program Office, named the NASA IVHM Technology Experiment for X-Vehicles (NITEX). NITEX was selected to fly on the X-34 reusable launch vehicle (RLV) developed by Orbital Sciences Corporation (OSC), and its research focused on the X-34 main propulsion system (MPS). After the X-34 program was cancelled, PITEX carried forward the previous research by building upon a prototype diagnostic system that was developed during the NITEX project.

The objectives and accomplishments of PITEX were achieved over a period of three years. This time was divided mainly between three phases of research and development: the Base Period, the Option 1 Period, and the Post-Option 1 Period. During the Base Period, the fundamentals of the diagnostic software system were developed and tested, and these results are summarized in reference 1. During the Option 1 Period, the diagnostic system was improved and enhanced by reducing the diagnostic time, improving the robustness to system variations such as sensor noise, biases, etc., and simplifying the diagnostic development process. This report focuses on the results from that development period along with a complete overview of PITEX. There was a short length of time after the Option 1 Period, called the Post-Option 1 Period, where limited development and testing was performed, and these results are presented in references 2 and 3.

During each phase of this project there have been a series of internal documents that have documented the objectives, progress, and results of the PITEX research and development efforts. The *Expansion Path* describes the underlying philosophy of PITEX and the development and implementation of the diagnostic software. The procedures and criteria that were used to test the diagnostic system are defined in the *Test Plan,* and the results from those tests are presented in the *Test Report*. Together, these three documents completely define the PITEX project and the work that was performed. This report is the culmination of those documents for Option 1.

This report is organized in the following manner. The next section provides an overview of the X-34 RLV; this includes descriptions of the vehicle, MPS, and nominal mission. After that, a summary of the PITEX diagnostic system and its components, the test application, and the overall research objectives (i.e., the *Expansion Path*) are presented. Next, a section for the test plan is presented, followed by the test results. General concluding remarks about PITEX finish the main body of the report. A list of references are included at the back of the report. The appendices at the end of the report provide an acronym list, supporting material, and supplementary test results.

2.0 X-34 Overview

The X-34 program was a joint effort by industry and government to develop, test, and operate a small, fully-reusable hypersonic flight vehicle that would demonstrate technologies and operating concepts applicable to future RLV systems. The primary organizations involved in this effort were Marshall Space Flight Center (MSFC) and OSC. MSFC was responsible for the design of the engine, and OSC was responsible for the MPS and flight program. In this section, background information on the X-34 is provided for the vehicle, main propulsion system, and nominal mission.

2.1 Vehicle

The X-34 is 58 ft long with a wingspan of approximately 28 ft. It had been designed to fly at Mach 8 at an altitude of 250,000 ft. A simple schematic of the main vehicle is shown in figure 1 (ref. 4). The X-34 was developed using design concepts and components from other proven launch systems. The specific details of every subsystem and component will not be addressed in this report, since a history of the vehicle and an overview of its main propulsion system can be found in the literature (refs. 4 and 5).

2.2 Main Propulsion System

The X-34 MPS and engine are responsible for providing the thrust that the RLV needs to meet the requirements of a mission. The engine is powered by LOX and RP-1 and the MPS provides for the loading, storing, delivering, and disposing of these propellants. Within the MPS, there are many subsystems that carry out these functions: the propellant tanks, the LOX feed system, the LOX fill and dump system, the RP-1 feed, fill, and dump system, the vent system, the pressurization system, and the pneumatic and purge system. The approximate location of each subsystem within the MPS is shown in figure 2 (ref. 5).

2.3 Nominal Mission

There are five phases during a nominal X-34 mission: pre-flight, captive carry, powered flight, post-flight, and landing. During pre-flight, the necessary ground operations are performed, such as filling the tanks with the propellants. After that, for captive carry, the X-34 is carried down the runway and up to the required launch altitude while attached to an L-1011 aircraft. Once captive carry is completed, the X-34 is released from the L-1011 and the engine is started. After powered flight has completed, the engine is shut down, and any excess propellants are dumped overboard. The X-34 then flies as a glider before it lands at a conventional runway. If a mission is terminated, the X-34 is designed to dump all propellants and still land safely. Figure 3 (ref. 5) depicts a nominal mission for the X-34.

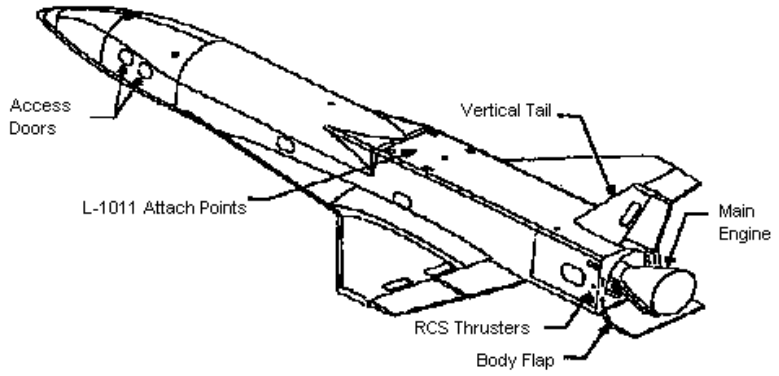

Figure 1.—X-34 vehicle outline.

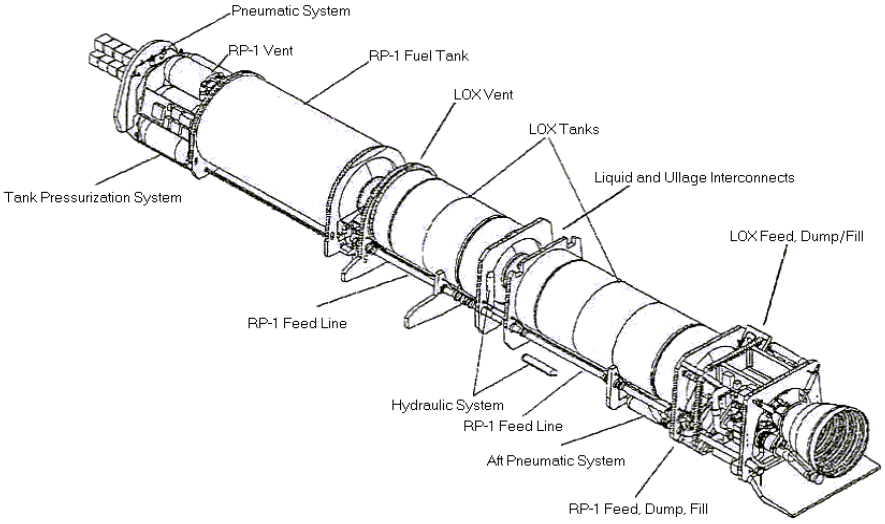

Figure 2.—X-34 MPS overview.

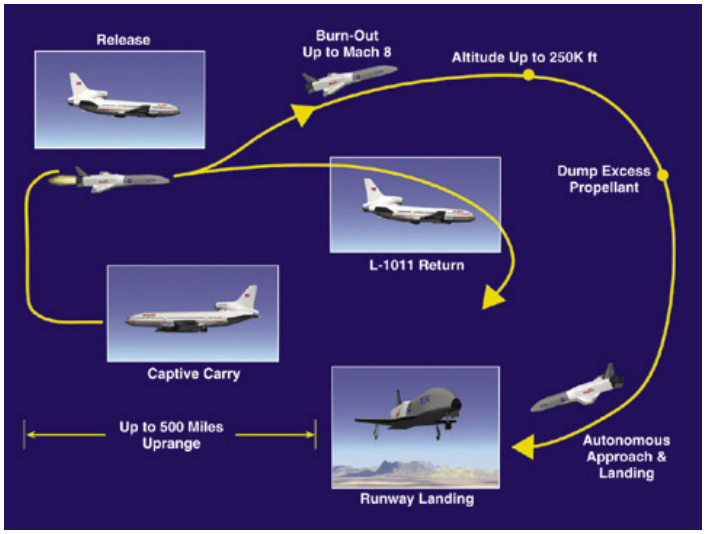

Figure 3.—Nominal X-34 mission.

3.0 PITEX Overview

The purpose of PITEX has been to advance the technology readiness level of real-time model-based diagnostics for propulsion subsystem applications. In particular, PITEX demonstrated the successful diagnosis of faults for the X-34 MPS by using a real-time diagnostic software system. This achievement was made possible through extensive collaboration between NASA Ames Research Center (ARC), GRC, and NASA Kennedy Space Flight Center (KSFC). This section provides an overview of the diagnostic system, the demonstration application, and the technology goals that were defined as the PITEX "Expansion Path."

3.1 Diagnostic System

Originally under NITEX, the diagnostic system was to encompass all phases of X-34 operation, including both real-time diagnostic and post flight analysis. The NITEX architecture provided for a real-time diagnostic package to be embedded onto the flight vehicle along with all supporting communication and ground-based elements. The PITEX demonstration system included the core modules of that NITEX real-time diagnostic architecture.

The PITEX system architecture for the diagnostic system is shown in figure 4. PITEX is an integrated software package that consists of a telemetry input system (TIS), monitors, real-time interface (RTI), Livingstone (L2), results output system (ROS), and ground processing unit (GPU). In addition, there is a virtual propulsion system that acts as a support component to provide simulated data for developing and testing the diagnostic system. Each component's functionality is described below.

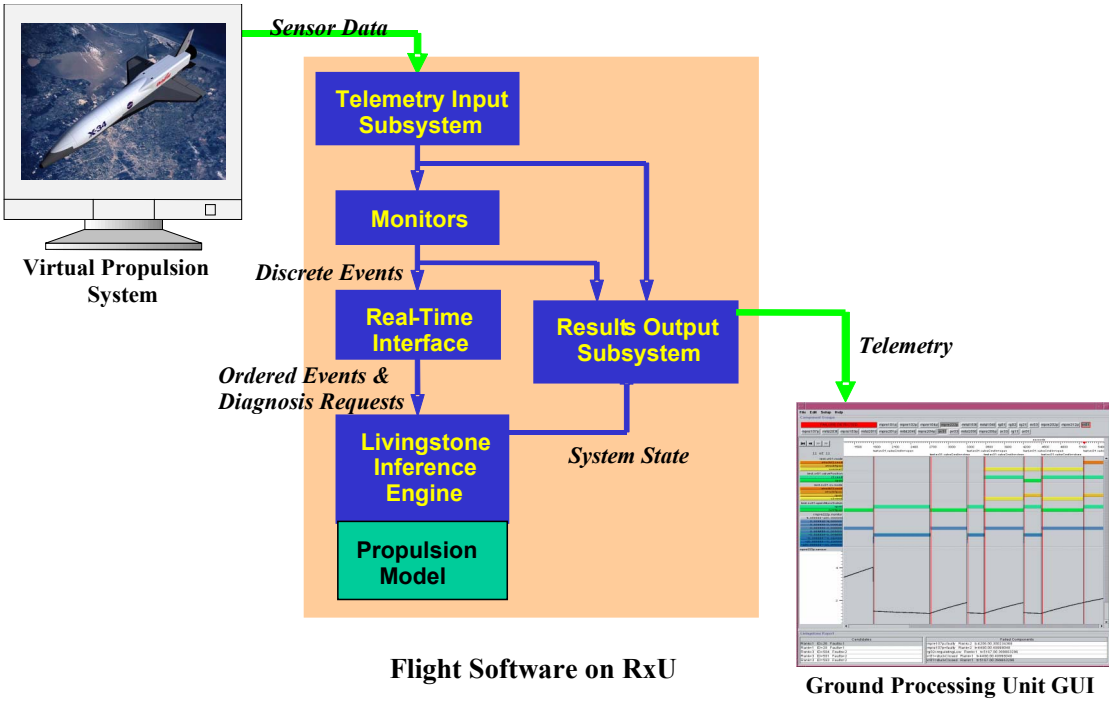

Figure 4.—PITEX demonstration architecture.

3.1.1 Telemetry input system

The TIS provides the interface between the flight-like data set and the Monitors, and it has three basic functions. First, the TIS provides an 80 msec timer to simulate the hardware interrupt that would occur when a frame of telemetry becomes available. Second, when the interrupt occurs, the TIS reads the next frame of data and stores it within an internal data buffer. Finally, access to the data buffer is provided to the diagnostic system by a TIS API routine. The TIS is designed to guarantee that the simulated telemetry data sets are available to the diagnostic system in real time.

3.1.2 Monitors

The Monitors convert raw sensor data into discrete events and pass that information on to the Real-Time Interface (RTI) component. The Monitors are a collection of routines, which retrieve the sensor data, perform any required processing, determine the position of the data in its parameter space and report the information to a standard message queue. The Monitors process both the binary discrete signals of the commands and switch indicators and the digitized performance sensors (e.g., pressure transducers). In addition, the monitors provide timing services made through request messages from the RTI.

3.1.3 Real-time interface

The purpose of the RTI is to transmit discrete events from the monitors to the Livingstone inference engine. To achieve this function, the RTI must handle four basic tasks. First, it translates the monitor information into a format that is understood by the Livingstone (L2) model. Second, it uses the timing information associated with the events to package the information into discrete Livingstone time steps. Third, it decides when to request a diagnosis from Livingstone. Finally, it dictates when Livingstone information is downlinked to the Ground Processing Unit (GPU) via the Results Output System (ROS).

3.1.4 Livingstone

Livingstone comprises a language for specifying a model of the system and a set of algorithms to use the models to track the system's state. A Livingstone model is composed of components (which may map onto physical components), connections between components, and constraints. A component is specified by variables, with a set of discrete, qualitative values for each variable in its local nominal and failure modes. For each mode, the model constrains how the components and the variables within the components are related to each other and how components can transition from one mode to another. The Livingstone inference engine receives data and diagnosis requests from the RTI. A propositional constraint system is formulated to track the state of the system over discrete time-steps choosing only those trajectories that are consistent with observations. In the event that discrepancies between the Livingstone model and the system observations occur, the diagnostic engine employs a Conflict-Directed Best-First Search (CBFS) strategy to determine the best probable candidates that would explain the conflict.

3.1.5 Results output system

The purpose of the ROS is to downlink diagnostic information via telemetry to the GPU. Monitors and Livingstone pass data messages to the ROS, specifying that the messages be either stored locally or downlinked. For the NITEX R1 demonstration, the ROS simulated the telemetry downlink by saving the data to a file on the local network. The GPU received the simulated telemetry by continuously reading the file and updating its state. During the PITEX demonstrations, the diagnostic telemetry was sent to the GPU via the internet. This data was then combined with the scenario data file, located locally on the GPU, to provide displays to the user.

3.1.6 Ground processing unit

The purpose of the GPU is to display diagnostic information from the on-board diagnostic system to ground personnel who would be monitoring the system. The GPU was designed for use by ground operators, who are primarily concerned with the status of the vehicle, and by diagnostic system developers who are interested in system development and testing. Therefore the GPU was designed with the idea that it should be relatively intuitive and easy to use, and yet provide enough information to understand why the diagnostic software is making a particular diagnosis.

For the NITEX project, the diagnostic system ran in a simulated VxWorks environment on a Sun workstation. Therefore, several of the components within that release of the GPU were mere placeholders for software that were required for handling and transmitting "real" telemetry. The placeholders were created so that the entire system could be run from end to end. In addition, there were facilities in place for archiving the data and replaying it at a later time. During the PITEX demonstrations, the diagnostic software was executed on a flight-like box, which then either saved diagnostic information locally in a log file or telemetered it to the GPU. There was still placeholder software on the GPU that performed functions such as decommutation of diagnostic and sensor data differently than what would be done on a completed flight system.

3.1.7 Virtual propulsion system

The Virtual Propulsion System is a support component that provides simulated data of the physical system for the purposes of providing insight into the behavior of the X-34 MPS feed system, which had not undergone system-level testing, and for verification and validation of the diagnostic software. This component includes routines that predict the behavior of various portions of the physical system during different modes of operation (i.e., propellant conditioning and bleed). These models produce output files with distinct parameters and sampling rates. Included in the virtual propulsion system component is a utility that combines the various output files into one flight-like data set. This utility adjusts the output data from the simulations to correspond to sensor locations, applies random noise to the data, and inserts the discrete signals (i.e., commands and switch indicators). It also adjusts the sampling rates output by the simulations to correspond to the expected telemetry rates. The final data set is in a standardized binary format, containing header information that records the file's content and creation date.

3.2 Application

Background information on the application for this work is described next; this section provides an overview of the test article, design reference mission, and numerical models that define the PITEX system application.

3.2.1 Test article

The original intent of the NITEX effort was to cover the entire X-34 MPS through all phases of the mission (launch preparation to post-flight). However, the initial implementation of the NITEX diagnostic system focused on the LOX conditioning portion of captive carry; this justified reducing the number of subsystems involved and provided a reasonable scope for the demonstration. The subsystems included were the following: pneumatic, pressurization, and LOX. Since the RP-1, purge, and reaction control system were not included, this restricted the number of components modeled and monitored while still offering unique processing challenges.

For the PITEX, application, the RP-1 subsystem was added in order to complete the modeling and representation of the captive carry phase mission. Shown in figure 5 is a schematic of the PITEX X-34 MPS, and each component is described in appendix 8.2. Some of the primary functions for the operating subsystems are as follows. The vent/relief system (ref. 6) prevents over-pressurization of the tanks, and it provides propellant conditioning for the LOX. Because there is no refueling of the X-34, the tanks provide for the LOX and RP-1 storage for a complete mission. The LOX and RP-1 feed systems (ref. 7) deliver propellant for engine bleed, preparation for launch, and engine operation.

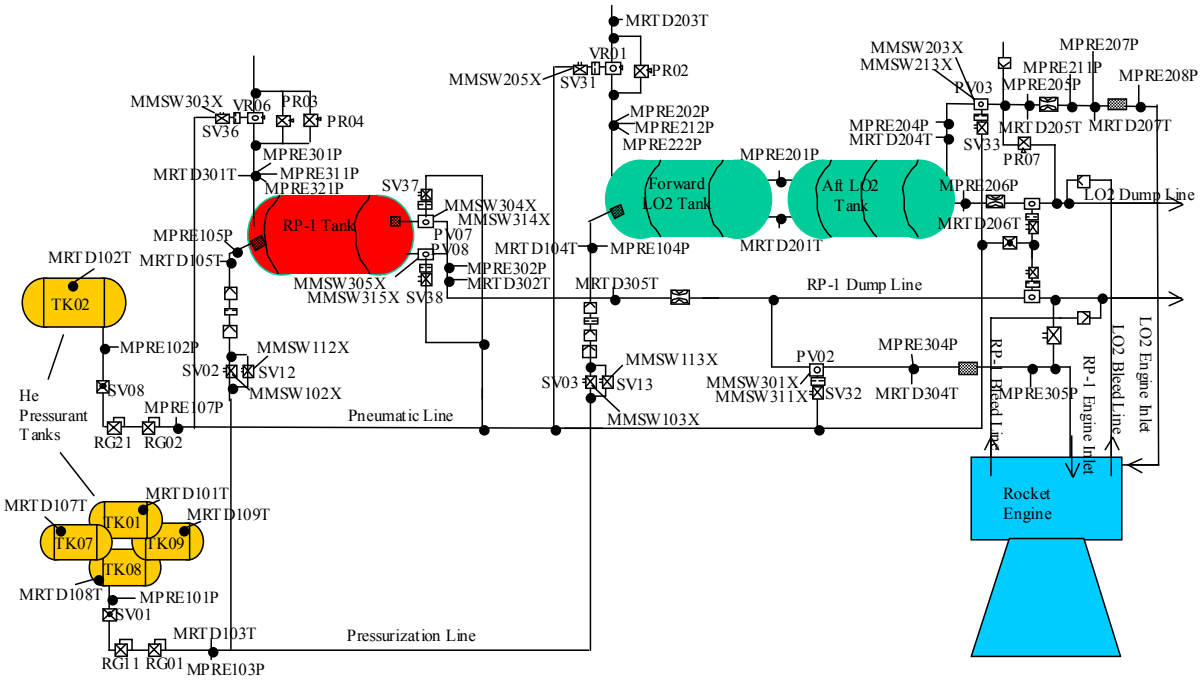

Figure 5.—PITEX X-34 main propulsion system schematic.

3.2.2 Design reference mission

Because the X-34 flight trajectory is divided into distinct phases, it was logical and practical to concentrate on one specific segment. As was mentioned earlier, captive carry was selected in order to decrease the scope of the project. In addition, this is one of the most important phases of the mission due to crew safety considerations of the piloted L-1011. During this phase of operation, the X-34 is carried to the required launch altitude of 38,000 ft while it is attached to the underside of the L-1011 aircraft. During the captive carry phase, the X-34 engine is not running, and most of the MPS subsystems are in a quasi-static state.

The timeline for the captive carry phase is divided into the following set of distinct events. Throughout the first half hour of captive carry, the MPS is locked-up. During this period, thresholds are selected for the vent/relief system and the pressurization system (ref. 8) so that they are inactive under nominal conditions. After this lock-up phase, the vent/relief system is activated to provide LOX conditioning. For 2 hr, this process maintains, within pre-defined thresholds, the nominal temperature and pressure in the LOX tanks. Once the L-1011, has reached the X-34's launch destination, the pressurization system is enabled to prepare the X-34 engine for flight. After that, RP-1 bleed (ref. 9) is performed for three minutes to ensure that all gases are removed from the engine fuel supply lines. Once that is finished, LOX chilldown and bleed (ref. 9) are performed, again to remove purge gases from the engine oxidizer supply lines and to thermally condition the engine component with the cryogenic fluid. Nominally, this process requires six minutes and is the last event to take place during captive carry. Figure 6 illustrates the captive carry timeline.

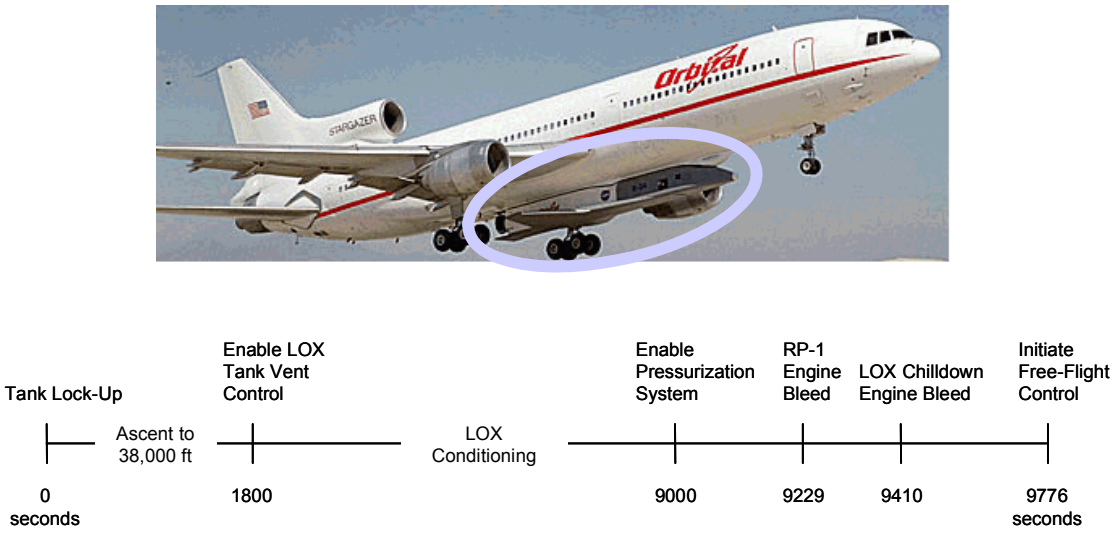

Figure 6.—Captive carry timeline.

3.2.3 Numerical MPS

The virtual propulsion system for the X-34 MPS includes both a Rocket Engine Transient Simulation (ROCETS) model and MATLAB routines that predict the behavior of various portions of the physical system during different modes of operation (i.e., propellant conditioning and engine bleed). The ROCETS program was selected because of its proven ability to reliably simulate large rocket propulsion systems. The physical scope of the model extends from the helium bottles to the LOX and RP-1 subsystems, which includes the propellant tanks, ullage venting systems, feed lines, and dump lines (as shown in fig. 5). The ROCETS model does not include the purge, reaction control, or pneumatic systems since it was initially developed to simulate delivery of propellants to a LOX/RP-1 engine during powered flight. As a result, it is not well-suited for simulating LOX conditioning since the dynamic behavior of the LOX tanks during this period is substantially different than its behavior during steady-state powered flight. Therefore, a MATLAB code was used to simulate LOX conditioning. The ROCETS/MATLAB models produce output files with time histories of selected parameters.

The numerical models provide the capability to study both nominal and off-nominal behavior of the X-34 MPS feed system. Several off-nominal scenarios have been simulated, and they are listed in appendix 8.3. These scenarios include valves sticking closed or open, valves spontaneously closing or opening, regulator failures, and sensor and microswitch failures. Subtle degradations such as the clogging of a filter, an obstruction in an orifice, or degradation in valve actuation can also be simulated.

Due to the fact that the accuracy of the data sets generated by the virtual MPS is dependent upon the approximations made when defining the components and the physical processes, there is some amount of modeling error present in the results. These types of errors can be substantially reduced by obtaining accurate component and system-level data and anchoring the simulations with these test data. Although this step in the diagnostic systems development process was not possible in the case of the X-34, the data generated by these numerical models permitted appropriate testing of the diagnostic software.

3.3 Expansion Path Objectives

As part of the development of this project's advanced health monitoring capabilities, technology goals were defined. They are listed below and collectively they defined the "Expansion Path."

- Demonstrate the Scalability of the System
- Demonstrate the Ability to Handle Sensor Noise

- Demonstrate the Ability to Handle Sensor Failures
- Demonstrate the Ability to Handle Telemetry Issues
- Provide Valve Timing Information
- Improve the Usability of the Ground Station
- Improve System Response
- Improve the Diagnostic System Development Process

During the course of the PITEX project, improvements and enhancements to the diagnostic software system were implemented that targeted these goals, and they are summarized below in table 1.

TABLE 1.—ENHANCEMENTS INCLUDED IN THE SYSTEM-LEVEL PITEX EXPANSION PATH TESTS

Module	Base period	Option 1
Monitors	• Expanded initialization and header files to include RP-1 components, redundant channel comparisons, threshold required for in-range sensor failure detection • Absolute delta monitor	• Implemented statistical rate monitors • Modified thresholds for IVTB scenario • Optimized monitor output • Capability to limit CPU usage
RTI	• Policy ties command to affected observations • Policy doesn't request diagnosis when observations represent uncertainty increase only • Policy only sends observations to Livingstone that differ from previous timestep	• Policy allows multiple diagnostic requests for each command issued • Policy allows a "find-fresh" of the diagnostic analysis instead of a "prune and search" • Settling times of observations are related only to events that can influence their values
Livingstone	• Model includes RP-1 components • Model capacity increased to handle microswitch faults and double (in-range) sensor failures	• Model includes pressurization indicator threshold • Implemented non-negative flow in pressurization valves • Reduced history that Livingstone tracks to length of one instead of three timesteps
GPU	• Availability of a schematic view	• Display downlink telemetry in real time • Ability to "browse" modules • Failures in subcomponents are displayed

Along with a description of each technology goal, these modifications are more fully described below.

3.3.1 Demonstrate the scalability of the system

The purpose of this Expansion Path activity was to assess any performance issues uncovered in expanding the coverage of the current diagnostic software and to track the flexibility of the current architecture in assimilating this expanded coverage. Expanding the coverage could include more physical components, more phases of operation, additional failure scenarios, or a combination of all three types.

During the Base Period, the expanded coverage involved the addition of the RP-1 tank and feed system during the captive carry mission phase. The rationale used to select which components to be monitored and modeled was the same as that used for the NITEX LOX component selection. Failure scenarios were selected based on fault severity, detectability, probability of occurrence and the capability

of the tools to simulate the fault. The combined LOX/RP-1 feed system was tested using simulated data for the nominal scenario and failure scenarios. Backward compatibility of the expanded model was ensured by correctly processing previous scenarios.

During Option 1, there were two activities in this Expansion Path category. The first was the successful diagnosis of an unresolved fault scenario from the Base Period, PFS7. Secondly, an additional scenario was added that enabled a demonstration of IVHM system-level communication. This scenario, designated IFS1, added limited fault detection capability to the LOX tanks.

3.3.2 Demonstrate the ability to handle sensor noise

The purpose of this Expansion Path activity was to determine the system's ability to perform in the presence of realistic sensor signal noise. Sensor signal noise added uncertainty to the information extracted from the signal.

During the Base Period, monitor data processing and RTI policy changes were implemented to compensate for potential sensor noise. These Base Period monitor and RTI policy enhancements were evaluated using select NITEX R1 failure scenarios. During Option 1, the statistical monitors developed in the Base Period were integrated and tested on the flight-like hardware.

3.3.3 Demonstrate the ability to handle sensor failures

The purpose of this Expansion Path activity was to improve the diagnostic system's ability to perform in the event of sensor failures. The key areas here are how the sensor failure detection is propagated through the diagnostic system and what impact certain sensor failures may have on specific fault scenarios.

During the Base Period, microswitch and control-loop sensor faults were incorporated. No activities were planned during Option 1.

3.3.4 Demonstrate the ability to handle telemetry issues

The purpose of this Expansion Path activity was to determine the system's ability to address telemetry issues. These issues included reduction in bandwidth, prioritization of the telemetry information, and telemetry data corruption and dropouts.

No Base Period activities were conducted in this Expansion Path category. During Option 1, compression of the telemetry data was implemented. Both the Livingstone and GPU modules were modified to generate and decode this compressed information. The compressed data format demonstrated improved performance by reducing the required output throughput for the software.

3.3.5 Demonstrate the ability to handle valve timing information

The purpose of this Expansion Path activity was to determine the system's ability to detect and report valve timing information. Valve timing assessment can provide information regarding expected future performance of a valve within the same mission or highlight the need for between-flight maintenance. Although the post-flight diagnostic assessment component of the NITEX architecture was not implemented, valve-timing capability would certainly be an important input to such ground-based analysis.

The Base Period applied and tested this monitor capability. No activities were planned during Option 1. However, the robustness data sets did incorporate slower valve effects for some selected components in the RP-1 subsystem.

3.3.6 Improve the usability of the ground station

The purpose of this Expansion Path activity was to advance the capabilities of the GPU. The advancements were designed to improve the usefulness of the system for personnel tasked with assessing the current health of the system. In addition the GPU could be used to further develop and verify the diagnostic system.

The primary Base Period enhancement was the addition of a schematic view of the test article. During Option 1, the enhancements included the following: downlink telemetry displayed in real time, ability to browse inside a module and the display of subcomponent failures in the schematic view.

3.3.7 Improve system response

The purpose of this Expansion Path activity was to improve the diagnostic system response. The system must be capable of providing fast and correct diagnoses of the system state.

Base Period efforts identified areas where improved system response is required. During Option 1, enhancements focused on reducing the response time from the statistical rate monitors without sacrificing the monitor's accuracy or robustness to signal noise. In addition, the RTI was also modified to provide quicker diagnostic responses to available observations.

3.3.8 Improve the diagnostic system development process

The purpose of this Expansion Path activity was to test and expand on the current validation and verification (V&V) practices applied to the PITEX effort. These techniques helped to standardize software development and provided more efficient testing of the system.

This Expansion Path category was added at the beginning of Option 1 as an outcome of the Base Period testing phase. During Option 1, enhancements concentrated on improving the robustness of the diagnostic system to nominal parameter variations, developing and implementing generic components in the diagnostic model, and investigating V&V techniques and tools to gauge and improve the robustness of the real-time software. In addition, code reviews of the monitors and RTI were performed.

4.0 Test Plan

This section describes the test plan that was implemented for evaluating the expanded capabilities of the PITEX diagnostic system. The system-level and component-level tests have been defined based on the PITEX Expansion Path objectives, and the procedures, that were followed to perform these tests, are described. In addition, the metrics collected to evaluate their performance are also included.

4.1 Test Software and Hardware

The testing of the PITEX diagnostic system during Option 1 was performed from December 5, 2002 to March 15, 2003. The hardware and software requirements for the implementation and operation of the diagnostic system are described in the following sections.

4.1.1 Hardware

The diagnostic software resided on a Radstone PPC4A-750 VME single Board Computer. The card was housed in a chassis with a VME backplane and SCSI hard drive. I/O ports provided both serial and ethernet accessibility to the card.

Table 2 describes the system configuration used for the system-level tests.

TABLE 2.—TEST SYSTEM CONFIGURATION

Software Version	PITEX 2.0
Host	Sun UltraSparc 60
Host OS version	Sun Solaris 8
Target	PPC603
Target OS version	VxWorks v. 5.4
Time slicing	Off
Timing method used	Hardware interrupt timer

The GPU hardware is a personal computer with a Pentium III 550 MHz processor, 256 MB of ram, and 30 GB of hard disk storage. The GPU used the Linux Operating System, Redhat version 6.2 or later, and communicated to the real-time Ground Unit through a TCP/IP ethernet connection.

4.1.2 Software

The PITEX diagnostic software was compiled in the Tornado II VxWorks (release 5.4) environment using the compiler supplied with Tornado. (A modified version of gcc release 2.7.2.) The VxWorks kernel was based on the Radstone board support package (release 1.2/1). The diagnostic system was based on the PITEX demonstration software, release 2.0, with Livingstone release 2.7.4. The configuration parameters used are recorded in table 3.

TABLE 3.—LIVINGSTONE CONFIGURATION PARAMETERS

Parameter	Value	Description
L2SearchMethod	CBFS	Conflict-Directed Best First Search method
L2MaxCBFSCandidates	10	Upper bound on the number of candidate diagnoses returned by a CBFS search
L2MaxCBFSSearchSpace	5000	Upper bound on the number of nodes searched for CBFS
L2MaxCBFSCutoffWeight	6	Upper bound placed on the number of candidates that differ in more than one time step for CBFS
L2MaxCoverCandidateRank	8	Upper bound on the rank of candidate diagnoses returned by a search
L2MaxHistorySteps	1	Upper bound on the number of time steps for which complete search information is kept
L2ProgressCmdType	Full	Type of search progression
L2NumTrajectoriesTracked	10	Maximum number of tracked candidates
L2FindCandidatesCmdType	Find-Fresh	All candidates cleared with each search

4.2 Performance Tests on Flight-Like Hardware

Performance of the PITEX software was measured by running tests (nominal and failure scenarios listed in appendix 8.3) on the flight-like hardware to determine the validity of diagnosis, resource utilization (CPU usage and memory requirements), and timing results. Except as noted in subsequent sections, all code was compiled with the following flags:

TABLE 4.—COMPILATION CONFIGURATION FOR HARDWARE TESTS

Flag	Description	Flag Source
-O3	Optimization level for gcc	gcc
-ansi	Sets compiler to use ANSI C	gcc
-nostdinc	the compiler will not search the standard directories for header files	gcc
-nostdlib	Don't link the standard libraries unless specifically passed to the linker	gcc
-fvolatile	Consider all memory references through pointers to be volatile	gcc
-f-no-builtin	Don't recognize built in functions that do not begin with two underscores.	gcc
-fno-defer-pop	Always pop function arguments as soon as the function returns.	gcc
-D_REENTRANT	Causes reentrant code to be generated	gcc
-DVXWORKS	For use in VxWorks applications	VxWorks
-DPOSIX_Q	Compiles in message queues used by POSIX. Otherwise, SYS_5 queues are used	PITEX
-DCPU=PPC603	Used by VXWORKS to build code for the PPC CPU.	VxWorks
-Wall	Display a default set of compiler warnings	gcc
-DRW_MULTI_THREAD	Specifies Rogue Wave multi-threading	gcc
-DINTIMER	Use a precise 80 ms timer in the TIS code	PITEX
-DENABLE_LOG_OUTPUT	Enables the output of debugging messages and TLM to log files	PITEX
-DTLM_OUT=LOGFILE	Sends all diagnostic telemetry to a log file. The default if contact with the GPU cannot be established and if logging is enabled.	PITEX
-DMEASURE=PITEX_NONE	Sets a standard PITEX run. Value may be changed for various PITEX tests (e.g., memory, CPU or IPC)	PITEX
-W	Enables the display of extra compiler warnings	gcc
-fno-rtti	Tells the compiler and linker to exclude Run Time Type Information from the executable code	gcc

4.2.1 Validation

When each test scenario was processed, a monitor log file, *monitor_debug.out*, and a ROS log file, *downlink.output,* were generated. A routine was developed that condensed the information from these files into a timeline layout of the system's diagnostic performance. The assigned fault candidates were analyzed for correctness based on the physical understanding of the system and the observations available at the time of the diagnosis. This test was executed with PITEX code compiled with the following additional flag: -DCEF_DEBUG, which enabled the output of extra information to the log files.

Metrics to be collected.—Due to extensive changes made to the PITEX code, the direct comparison of Option 1 Expansion Path test results to earlier test results, by automated means, was not feasible. Therefore, the bulk of the validation testing was performed manually.

A correct diagnosis included the following:

- The injected fault shall be among the fault candidates reported.
- All candidates in this diagnosis are explainable given the evidence available.

In addition, the following acceptance criteria were used:

- A correct diagnosis is obtained some time after the fault is injected.
- Once the injected fault is diagnosed, it remains in the list of candidates until conclusion of the test.

4.2.2 CPU utilization

The VxWorks "Spy" utility was used to determine task execution times at a rate of 1000 ticks per second, and this information was recorded in a log file. The recorded data was then divided by 1000 to obtain the CPU utilization as a percentage of time for each one-second time period.

The test code was compiled with changes to the following flags:

Flag	Description
-DMEASURE = PITEX_CPU	Includes code required to collect CPU usage metrics
-DDISABLE_LOG_OUTPUT	Disables all informational logging output except for error messages
-DTLM = NONE	Disables all telemetry output to either the GPU or a local log file

Metrics to be collected.—Average CPU usage over the entire scenario was computed. The average was determined for each task and summarized for the overall software system on a per/test basis. The usage was measured as a percentage of the maximum CPU capacity.

4.2.3 Memory utilization

Static, stack, and dynamic memory was monitored during memory utilization tests in the following manner. Static memory usage was recorded for each object module downloaded to the PowerPC target through the use of VxWorks tools. The maximum amount of stack memory usage was recorded by using the VxWorks "checkStack" routine just prior to task expiration. Custom routines were written to track dynamic memory allocation on a per task basis using memory partitions. When a PITEX task dynamically allocated memory, the memory was allocated from its own memory partition using the custom routines. A data collection routine then determined how much memory was allocated for the partition.

The test code was compiled with changes to the following flags:

Flag	Description
-DMEASURE = PITEX_MEM	Includes PITEX code which enables the collection of data for memory usage
-DTLM_OUT = NONE	Disables Telemetry output (including to the log file) from PITEX
-DDISABLE_LOG_OUTPUT	Disables all log output except for error messages

Metrics to be collected.—The objective of these tests was to determine the memory requirements for PITEX software. This information can then be utilized to infer memory requirements when expanding the software for a given application or to cover a new application. There is no "pass/fail" requirement for this test other than total maximum memory usage for all tasks shall not surpass the available memory on the target. Data collection for the test was made via VxWorks tools.

4.2.4 Timing

Timing analysis required that the diagnostic time be extracted from the simulated telemetry file. The diagnostic time was determined by reading the last ROS frame update timestamp in the telemetry file prior to the Livingstone diagnosis block that corresponded to the fault.

The test code was compiled with the addition of the following flag:

Flag	Description
-DPRINT_TMR_EXP_MSG	Enables PITEX code to print the timer expiration times to the telemetry log file.

Metrics to be collected.—The diagnostic log file, *downlink.output,* contained data from which diagnostic performance parameters could be determined for a specific scenario. The times of interest were defined as follows:

- To—observable time. This is the time at which the fault could be sensed.
- Tl—fault diagnosis time. This is the time at which Livingstone reported the correct diagnosis as a candidate.

Given the above definitions, the following performance parameter was measured once for a specific failure scenario, at the time the failure occurred.

- Diagnostic delay = Tl – To. This is the overall delay for the system.

In addition, the following aggregate metrics were used to indicate performance over the suite of validated scenarios:

- Maximum and average diagnostic delay - the average diagnostic latency provides the Mean Time to Failure Detection.

4.3 System Robustness Tests

The robustness of the diagnostic system to nominal parameter uncertainty was evaluated with this set of tests. This type of uncertainty was selected because, in real life, build-to-build variations will cause a system to behave differently. The diagnostic system must be able to tolerate these variations without any impact to its performance. Parameters in the RP-1 subsystem were selected based on information obtained from Orbital Sciences Corporation. The selected parameters included the following: setpoint for the primary regulator in the pressurization system (RG11), opening time for the RP-1 tank feed pneumatic valve (PV07), opening time for the RP-1 feed pneumatic valve (PV02), and opening/closing times for the RP-1 tank primary pressurization valve (SV02). Their nominal variation is shown in table 5.

TABLE 5.—NOMINAL VARIATION OF SELECTED X-34 MPS PARAMETERS

Parameter	Maximum value	Minimum value
RG11 Setpoint (psia)	350 +5 percent	350 –5 percent
PV07 Opening (sec)	3	0.3
PV02 Opening (sec)	3	0.3
SV02 Opening/Closing (sec)	0.05	0.025

Metrics to be collected.—Table 17 in appendix 9.4 lists a set of tests for these parameters when they were varied by their maximum and minimum values. The simulated data sets for these tests were processed by the diagnostic software. However, since this was the first time that these tests had been performed, comparisons could not be made with previous results. A successful test verified that the PITEX diagnosis was “nominal.” In other words, there were no detected faults for any of the tests.

4.4 CPU Restriction Tests

Average CPU usage was measured as one of the computational performance characteristics of the diagnostic software. As part of the Eastern and Western Range safety requirements, it is necessary to guarantee the performance of the diagnostic software even with the restriction of computer resources (ref. 10). Therefore, a method for bounding the CPU usage was implemented by causing the software to “sleep” for a calculated duration in between each telemetry frame. Evaluation of this CPU restriction determined the impact on the diagnostic delay and the PITEX diagnostic software’s ability to perform under adverse computational conditions. These tests were performed on fault scenarios NFS3, NFS9, and NFS18.

Metrics to be collected.—The same test article and procedures that were defined for the performance tests were used here, except that the maximum amount of CPU available to the diagnostic system was varied. Maximum CPU usage was limited to 5, 10, 25, and 50 percent, and it was set via the command line prior to initiating the test. For each test, the diagnostic delay (timing), as a function of available CPU, was determined and the diagnostic results verified.

4.5 GPU Operations Tests

The GPU GUI was started on the GPU hardware, and tests were performed using PFS7 and IFS1 test results. The test article and procedures defined for the performance tests were the baseline for the GPU tests. Exceptions to this were the following:

- The code was compiled using the flag –DTLM_OUT = TLM_STREAM.
- The GPU IP address was specified on the command line at the start of the test.

Metrics to be collected.—Verification that the GPU was displaying the correct diagnostic information was the main metric collected. In addition, GPU verification included that, in the graphic view, the module that contained the failure was properly highlighted, when it was implicated.

4.6 Diagnostic Development Tests

Another Livingstone diagnostic model of the X-34 MPS had been developed using generic components. Diagnostic development tests were conducted on this generic model, and these tests helped evaluate the use of generic components for the following: sensors, valves, and regulators. Validation testing was performed on the generic model, as described in section 4.2. In addition, hardware performance tests for CPU and memory utilization as well as timing were conducted for the nominal scenario and NFS18.

Metrics collected.—Generic component testing was verified by checking that the correct diagnosis was obtained and comparing it with the validation results of the non-generic PITEX model. For the hardware performance tests, the resource utilization and timing information were also compared with the test results of the non-generic model.

5.0 Test Results

This section summarizes the test results corresponding to the Expansion Path activities and associated unit and system-level tests as outlined in section 4. These tests were aimed at demonstrating and measuring the performance of the PITEX Option 1 diagnostic system enhancements. The Option 1 enhancements focused on five main areas: scalability, telemetry issues, improving the GPU GUI, improving the system response, and improving the diagnostic system development process.

5.1 Validation

The validation tests were conducted on all available scenarios: 24 failure scenarios and one nominal scenario. Upon completion of each validation test, two log files, *downlink.output* and *monitor_debug.out*, were used to determine the diagnostic result and verify the PITEX response. Post-processing of the *downlink.output* file had to be done to translate the fault diagnostic information into human readable format. A utility routine combined and consolidated the pertinent information from the two log files into a diagnostic summary file. Diagnostic time values were extracted from log files generated during the timing tests, because they represented a more accurate time value without the additional logging overhead.

For each scenario, the diagnosis and acceptance criteria have been met, as defined in section 4.2.1. appendix 8.5 contains a tabulated summary of the validation results.

5.2 CPU Utilization

The PITEX code and VxWorks kernel were compiled using the GNU compiler set for optimization level 3 and PITEX logging turned off to ensure that debugging information would not adversely affect the results. In addition, the VxWorks *Spy* utility was used to measure per-task CPU usage; therefore, it was included in the VxWorks kernel during build time.

A function was written to initialize the *Spy* interrupt clock at 1000 ticks per second and to collect data at one-second intervals. The priority level for this task was set between the level of the monitor routines and the data collection routines. The task's output was directed to a file (*spy.log*) on the PowerPC target. Upon completion of the test, the report file was downloaded to the host computer and parsed with a Tool Command Language (Tcl) script to extract the CPU usage data for PITEX tasks. The data was then saved to a Microsoft Excel (Microsoft Corporation) file.

The data collected represents the execution time for each task, measured in clock ticks. This data was turned into a percentage of CPU time on a task-by-task basis. The overall CPU usage time by the PITEX code was also calculated.

Two problems occurred with collection of the CPU usage metrics. First, the original resolution was found to be too low, and CPU usage rates for many of the PITEX tasks significantly varied from scenario to scenario. A resolution of 1000 ticks per second was chosen, because it was the slowest sample rate that yielded consistent CPU usage times for the PITEX modules. Increasing the rate further would have impacted the performance of PITEX by shifting the available CPU time from PITEX to the CPU data collection code itself.

The second problem was that maximum Livingstone CPU usage had a duration of one second or more. Since the sample rate was one second, there were occurrences where maximum Livingstone CPU utilization crossed into two sampling periods. As a result, the maximum usage was calculated to be lower than the actual value. An attempt to increase the sample period to 2 sec was made, but the maximum Livingstone usage values were still suspect. A further study of the problem was deferred but never revisited before the project was cancelled.

Table 6 summarizes the CPU usage results for PFS8.

TABLE 6.—BREAKDOWN OF PFS8 CPU USAGE

Tasks	Maximum %CPU usage	Average %CPU usage
Livingstone/RTI	92.8	0.56
Monitors	3.1	2.34
Data I/O	5.5	0.17

Figure 7 illustrates the increase in CPU usage in PFS8 due to the injected fault at 9000 sec. As the figure shows, average CPU usage is very low (approximately 2.4 to 2.97 percent), but occasionally usage significantly increases. These increases occur each time Livingstone performs a diagnosis regardless of the scenarios tested. For this scenario, Livingstone performs a diagnosis approximately 10 sec after the fault injection time. This ten second delay occurs to allow system transients to settle. Livingstone then performs a diagnosis by searching for failure candidates and an initial spike in CPU usage can be seen. After 9200 sec, the spikes in PFS8 become even higher, due to Livingstone tracking multiple hypotheses. This CPU usage spike is absent for the nominal scenario as shown in figure 8.

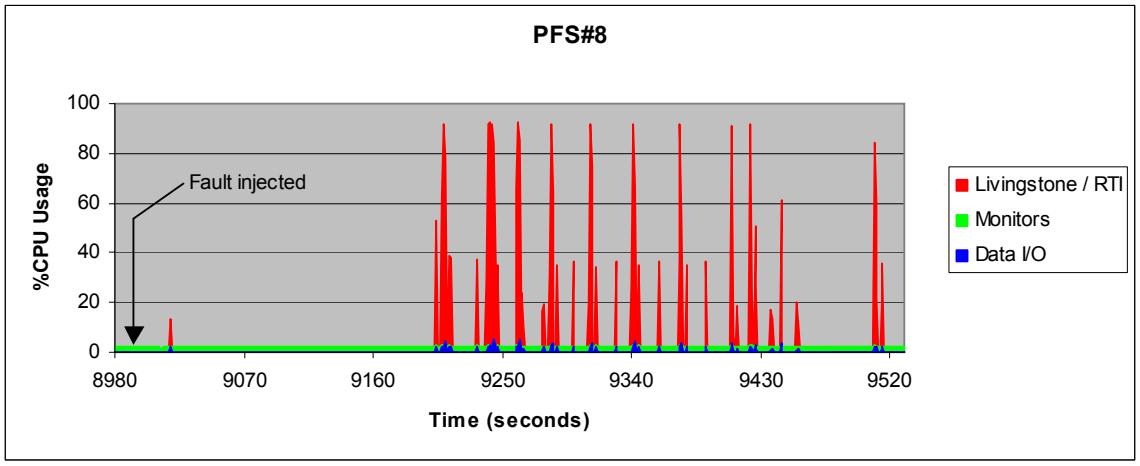

Figure 7.—Total CPU usage for PFS8.

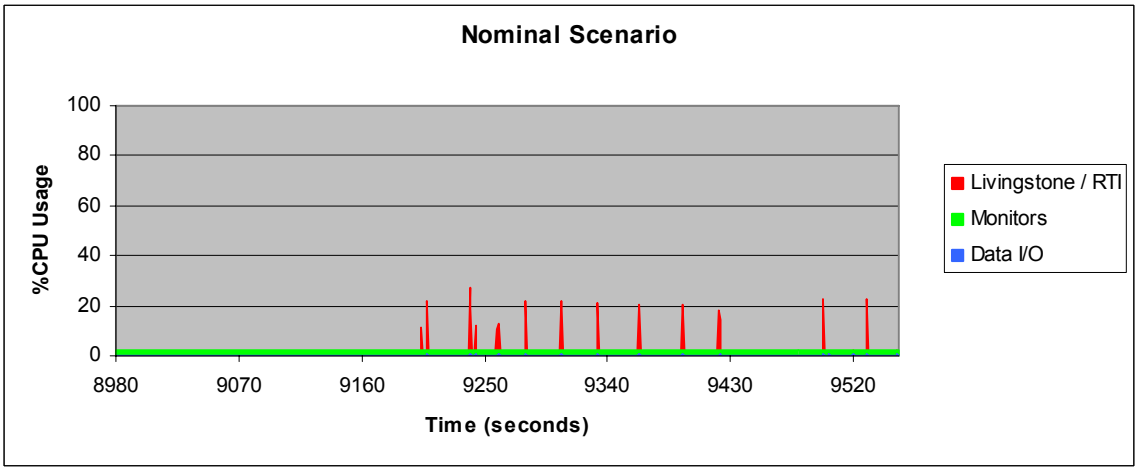

Figure 8.—Total CPU usage for the nominal scenario.

Table 7 summarizes the maximum and the average CPU usage as a percent of the maximum CPU capacity for all of the scenarios. The values in this table are summed up across all of the tasks. Among these tasks, the monitors account for the bulk of the average CPU usage (e.g., see table 6, since they run continuously, scanning sensor data for discrete events. However, they are designed to process events quickly, and therefore their maximum CPU usage never exceeds 3.07 percent. Livingstone, on the other hand, runs infrequently. Thus, the impact of its processing on the average value is minimal. However when it does execute, it uses a significant number of processor cycles and is the main contributor to the maximum usage values.

Most of the scenarios show a high maximum CPU usage. It should be noted that these maximum values correspond to spikes in CPU usage that are approximately 1 second in width. These spikes, as mentioned above, are due to Livingstone activities. However, the high numbers are not a concern; since, Livingstone has a low execution priority and will not block other time critical processes from running.

TABLE 7.—CPU UTILIZATION SUMMARY RESULTS

Scenario	Maximum %CPU usage	Average %CPU usage
Nominal	33.63	2.53
NFS1	39.4	2.45
NFS3	46.9	2.50
NFS6	24.0	2.43
NFS7	25.3	2.46
NFS9	50.7	2.44
NFS10	33.8	2.48
NFS11	21.9	2.46
NFS12	20.9	2.46
NFS13	36.4	2.44
NFS14	98.1	2.96
NFS16	97.4	2.59
NFS18	97.4	2.60
NFS19	46.0	2.48
PFS1	93.1	2.63
PFS2	49.7	2.61
PFS3	29.3	2.46
PFS4	99.6	2.58
PFS5	46.3	2.46
PFS6	90.4	2.50
PFS7	29.0	2.56
PFS8	97.5	3.07
PFS9	97.0	2.57
PFS10	97.0	2.97
IFS1	81.1	2.62

Maximum percent CPU usage results should be considered approximations, as occasionally a spike in CPU usage is split up between two adjacent windows, due to the real-time nature of the measurements.

5.3 Memory Utilization

Three types of memory usage were examined during these tests: static, stack, and dynamic.

Static memory.—The static memory usage was recorded for each object module downloaded to the PowerPC via the VxWorks Browser tool. When source code is compiled for the VxWorks operating system, it is translated into executable machine code, which can be divided into the following three categories: “Text,” “Data,” and “Bss.” Table 8 presents static memory usage results for each of these categories.

The “Text” column shows the number of bytes needed to store processor commands. The “Data” column shows the number of bytes used by the variables that have been assigned a value. The “Bss”

column is the number of bytes for the block storage space (Bss), which is used as a variable "scratch pad." The sum of values in these three columns gives the amount of memory required for the program to be run from RAM. To determine how much space is required to store the program in a long-term storage device (e.g., EEPROM or NVRAM); only the sums of the "text" and "data" columns need to be calculated.

TABLE 8.—STATIC MEMORY RESULTS

	Text (bytes)	Data (bytes)	Bss (bytes)	Sum (bytes)
Utilities/IO routines	16,256	624	16	16,896
Livingstone	595,136	800	1,048,732	1,644,668
Monitors	37,136	576	0	37,712
RTI	87,616	5,360	3,648	96,624

Stack memory.—The maximum amount of stack memory usage was recorded by using the VxWorks *checkStack* routine just before task expiration. The results are collected in table 19 of appendix 8.6, and they show the number of bytes that each task used on the stack during each of the scenarios. The "Size" column shows the maximum amount of stack space that can be used for the task. That number is set when the task is spawned, and the value shown is the amount of memory available to the task once the VxWorks kernel overhead is subtracted. The "High" column shows the maximum amount of stack space that a task actually used during its execution.

As seen in the "Margin" column, there is a significant amount of excess memory allocated for utilization by the stack. The stack size could instead be set to 1 kB above the highest value recorded in order to save memory while allowing for a small buffer zone.

The RTI task has the highest stack usage. For the Base Period of PITEX, the RTI variables that had previously been dynamically allocated from the heap were moved to the stack. This removed the need for most of dynamic memory allocation (malloc) calls, thus increasing the execution speed and making the program more robust. The RTI task also had the highest stack margin. This was done to allow ample room to experiment with the buffer sizes for observations without having to change the stack size for each run.

Dynamic memory.—The dynamic memory allocation functions, such as *malloc, calloc*, etc., were overridden with custom memory routines that kept track of dynamic memory allocation on a per task basis. An initialization routine was created to establish a memory partition for each PITEX task from the system memory pool. When a PITEX task dynamically allocated memory, it is allocated from its own memory partition. This enabled the use of a single function call to determine how much memory was allocated for the partition. During any call to a memory allocation routine, the maximum amount of memory that was allocated for the current task was recorded. Another task, specifically used for memory utilization tests, saved the maximum memory recorded in a Microsoft Excel spreadsheet in 1 sec intervals. Once saved, the current maximum memory amount value was reset by the memory measurement task.

Table 9 shows the dynamic memory results. The Livingstone task is the largest memory user, because it allocates memory in 1 MB chunks even if it needs only a couple of bytes to perform its calculations. Conversely, the RTI task does not require any since, as described above, it uses stack memory for all its variables. The remaining processes use only minimal amounts of dynamic memory that vary slightly and are not significantly impacted by the failure scenario diagnostic requirements.

Finally, memory utilization was also assessed with respect to the total amount of system memory available. The PowerPC architecture limits its relative addressing to 24-bit offsets (i.e., the maximum memory size supported by the kernel is 32 MB). The memory size is automatically limited to 32 MB by the function sysMemTop. Any additional memory will be mapped, but it will not be accessible by the VxWorks kernel (e.g., malloc), download from code, etc.). This means that the maximum amount allocated for the runtime software (including the VxWorks kernel itself) is around 30.4 percent, as shown in figure 9.

TABLE 9.—DYNAMIC MEMORY RESULTS

Scenario	Livingstone (bytes)	RTI (bytes)	Monitors				Utilities/IO
			Command (bytes)	Set point (bytes)	Valve timer (bytes)	Timer (bytes)	TIS (bytes)
Nominal	1048680	0	5144	98248	19616	320	123104
NFS1	1048776	0	5144	98248	19616	320	123104
NFS3	1048776	0	5144	98248	19616	320	123104
NFS6	1048752	0	5144	98248	19616	280	123104
NFS7	1048752	0	5144	98248	19616	280	123104
NFS9	1048728	0	5144	98248	19616	280	123104
NFS10	1048728	0	5144	98248	19616	280	123104
NFS11	1048728	0	5144	98248	19616	280	123104
NFS12	1048728	0	5144	98248	19616	280	123104
NFS13	1048752	0	5144	98248	19616	280	123104
NFS14	1048848	0	5144	98248	19616	360	123104
NFS16	1048896	0	5144	98248	19616	360	123104
NFS18	1048800	0	5144	98248	19616	440	123104
NFS19	1048728	0	5144	98248	19616	320	123104
PFS1	1048752	0	5144	98248	19616	320	123104
PFS2	1048728	0	5144	98248	19616	320	123104
PFS3	1048800	0	5144	98248	19616	320	123104
PFS4	2097360	0	5144	98248	19616	320	123104
PFS5	1048728	0	5144	98248	19616	320	123104
PFS6	1048800	0	5144	98248	19616	320	123104
PFS7	1048728	0	5144	98248	19616	280	123104
PFS8	1048896	0	5144	98248	19616	360	123104
PFS9	1048896	0	5144	98248	19616	280	123104
PFS10	1048896	0	5144	98248	19616	280	123104
IFS1	1048752	0	5144	98248	19616	320	123104

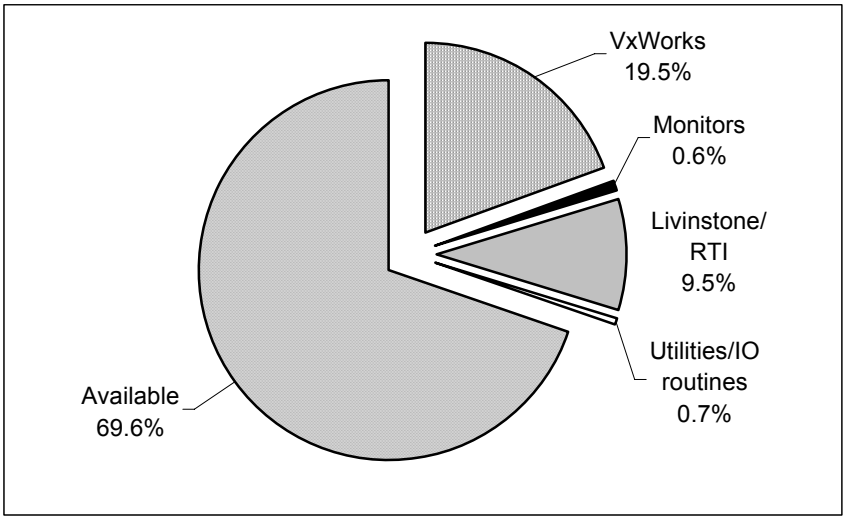

Figure 9.—Memory usage summary.

5.4 Timing

The information needed for this set of measurements was extracted from the *downlink.output* file, which was generated each time a scenario was run. With the exception of the output to the *downlink.output*, all debug logging was turned off. Table 10 contains the following key data points extracted from the *downlink.output* file:

- To—observable time.
- Tl—fault diagnosis time.

As defined earlier, diagnostic delay can be calculated as follows:

- Diagnostic delay = Tl – To.

In addition, the maximum and average diagnostic delays are listed to indicate performance over the entire suite of failure scenarios.

TABLE 10.—HARDWARE TIMING RESULTS

Scenario	To (sec)	Tl (sec)	Diagnostic delay (sec)
NFS1	9410.000	9413.200	3.200
NFS3	9539.790	9544.400	4.610
NFS6	2706.900	2710.000	3.100
NFS7	5000.000	5010.080	10.080
NFS9	2706.900	2712.960	6.060
NFS10	5000.000	5010.800	10.800
NFS11	5167.400	5170.480	3.080
NFS12	3331.000	3344.720	13.720
NFS13	5167.400	5174.560	7.160
NFS14	9000.000	9010.320	10.320
NFS16	9642.670	9645.920	3.250
NFS18	9735.290	9738.240	2.950
NFS19	9766.050	9769.200	3.150
PFS1	3301.700	3304.800	3.100
PFS2	9410.000	9413.440	3.440
PFS3	9359.000	9369.280	10.280
PFS4	9379.000	9406.080	27.080
PFS5	9383.86	9386.960	3.100
PFS6	9384.710	9387.040	2.330
PFS7	3301.700	3304.800	3.100
PFS8	9000.000	9010.320	10.320
PFS9	0.000	10.000	10.000
PFS10	0.000	10.000	10.000
IFS1	9199.000	9235.440	36.440
		Average diagnostic delay	8.361
		Maximum diagnostic delay	36.440

With the exception of PFS4, the diagnostic delay was decreased or remained the same for all of the fault scenarios (see appendix 8.4 for previous results). The competing processes of the failed component venting the tank and the control system continuously re-pressurizing the tank, occurring in the PFS4 scenario, actually confound the ambiguity of the observations and require a longer settling time to stabilize than either process individually. For the Base Period, the additional stabilizing time fell within the latency time period used by that version of PITEX and therefore allowed that version to report the fault earlier than the current version.

5.5 System Robustness

Since parameter variation information was obtained later in the Option 1 period, the monitor limits and diagnostic model used in PITEX release 2.0 did not reflect the variations listed in table 5. Therefore, PITEX was not expected to handle the complete set of robustness tests that are listed in table 17, without false reports being made. This expectation was confirmed when initial testing showed that false reports were generated for robustness tests 1 through 16. These tests contain variations in the primary pressurization regulator (RG11) setpoint value. Robustness tests 17 through 23 passed through the diagnostic system without generating any false reports.

Based on the initial test results, diagnostic software modifications were implemented. Specifically, adjustments were made to the threshold values in the monitor initialization file and in the diagnostic model. The 23 robustness data sets were retested using the modified software, and no false reports were generated. In addition, regression testing was performed on all the NITEX, PITEX, and IVTB fault scenarios using the modified software. Only one fault scenario, NFS14, had diagnostic results that differed from the validation test results. In this fault scenario, the primary pressurization regulator (RG11) is failed high at 370 psia, which is just at the upper threshold of nominal operation for this component (370 psia). With the addition of simulated noise to the signal, the component response can intermittently move back into and out of the nominal range of operation. As a result, PITEX alternated between reporting the component as not failed/failed. The other pressurization regulator failures (NFS16 and PFS8) are not within the proximity of the nominal range of operation. Therefore, their faults were detected without any problems.

5.6 CPU Restriction

For the CPU restriction tests, the amount of CPU available to the PITEX software was restricted to levels from 100 to 5 percent, and the amount of diagnostic delay for each fault scenario tested was recorded. During testing, an apparent timing conflict was discovered between the TIS functions and the VxWorks interrupt service routine.

These TIS functions and the interrupt service routine share the same two variables, *tis_ready* and *tis_update,* which are used as indices into the telemetry frame buffer. The interrupt service routine would write to these two variables while TIS functions were trying to read them. If the TIS functions could not read and use the two variables fast enough, before the interrupt service routine changed them, it resulted in wrong monitor data and, subsequently, duplicate ROS frames.

A temporary fix was devised to reduce occurrence of “race conditions” between the interrupt service routine and the TIS functions. The two variables were put into a cache so that the interrupt service routine could change the variables directly, and the TIS functions could read the variables from the cache. Although, this temporary fix did not correct the problem, it did reduce the number of frame duplicates by more than 95 percent. That was enough to ensure the correct diagnoses, even during severe CPU restriction testing, and to chart the diagnostic delays for each level of CPU restriction.

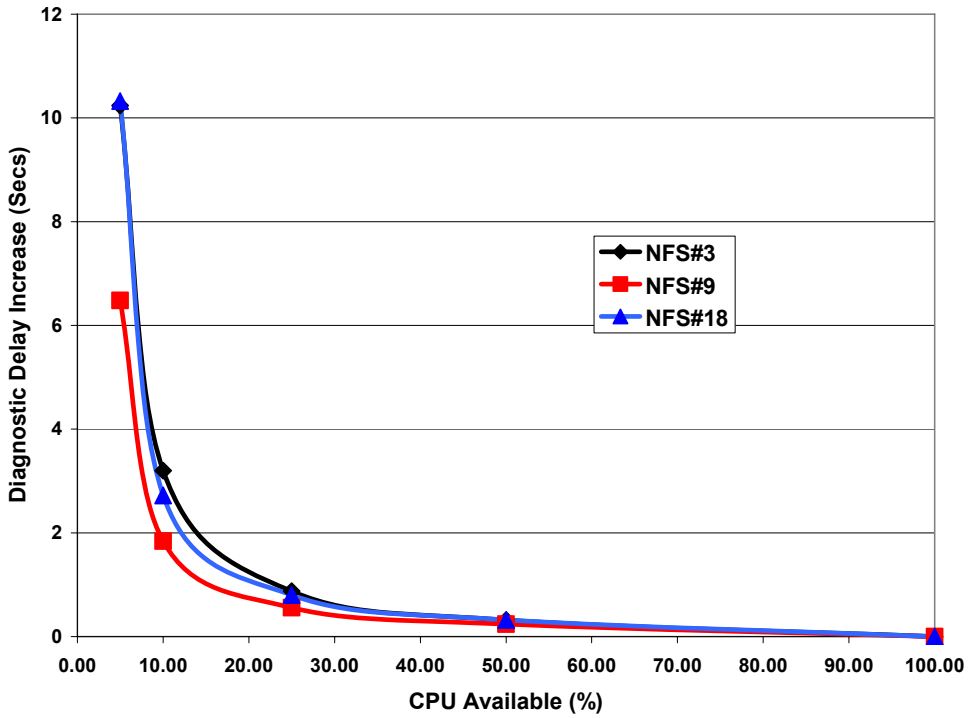

Figure 10.—PITEX diagnostic delay relative to 100 percent available CPU.

For each of the three fault scenarios tested, the correct diagnosis was reported by PITEX at each CPU restriction level. The diagnostic results from the CPU restriction tests match those from the validation testing, which are reported in appendix 8.5. The diagnostic delay due to restricted CPU is listed in table 11. Figure 10 illustrates the increasing diagnostic delay relative to 100 percent CPU availability, as the CPU restriction increases. In each case, the PITEX diagnostic software performance degraded gracefully, meaning that the PITEX diagnosis was completed and accurate.

TABLE 11.—CPU RESTRICTION RESULTS

Scenario	5% available	10% available	25% available	50% available	100% available
NFS3 diagnostic delay (sec)	14.848	7.808	5.488	4.928	4.610
NFS9 diagnostic delay (sec)	12.540	7.900	6.620	6.300	6.060
NFS18 diagnostic delay (sec)	13.272	5.672	3.752	3.272	2.950

5.7 GPU Operations

The GPU is used to display diagnostic information, from either the Real-time Flight Unit (RFU) or the Real-time Ground Unit (RGU), to ground personnel and diagnostic system developers. Therefore, the GPU GUI has been designed to provide diagnostic information of the system and verification information about the diagnostic results. During Option 1, the primary changes to the GPU were the following:

- Downlink telemetry displayed in real time.
- Browse inside a module in the schematic view.
- Subcomponents failures displayed in the schematic view.

The GPU testing was conducted on fault scenarios PFS7 and IFS1. From the test plan, two key items were identified as test criteria:

- Verify that the GPU is displaying the correct data
- Verify that in the graphic view the module that contains the failure is highlighted when it is implicated

Both fault scenarios passed the test criteria, and appendix 8.8 contains figures that display the GUI results for the two fault scenarios.

5.8 Diagnostic Development

There were no discrepancies in the validation results, and the diagnostic results match those reported in appendix 8.8. Hardware test results for the nominal scenario and NFS18 are shown below.

CPU usage

The generic model CPU usage results are presented in table 12. The maximum and average CPU usage are slightly varied when compared with the results in table 7.

TABLE 12.—CPU USAGE RESULTS WITH THE GENERIC MODEL

Scenario	Maximum %CPU usage	Average %CPU usage
Nominal	47.08	2.48
NFS18	97.55	2.65

Stack memory

The generic model stack memory results for the nominal scenario and NFS18 are shown in table 13.

TABLE 13.—GENERIC MODEL STACK MEMORY RESULTS

Task	Stack memory (bytes)		
		Nominal	NFS18
TIS	Size	9,728	9,728
	High	2,368	2,368
	Margin	7,360	7,360
LogTask	Size	9,728	9,728
	High	2,928	2,928
	Margin	6,800	6,800
ROS	Size	19,728	19,728
	High	5,856	5,856
	Margin	13,872	13,872
TimerMonitor	Size	9,720	9,720
	High	2,816	2,816
	Margin	6,904	6,904
CmdMonitor	Size	9,720	9,720
	High	2,832	2,728
	Margin	6,888	6,992
SetPointMonitor	Size	14,720	14,720
	High	6,904	6,904
	Margin	7,816	7,816
vlvTimerMonitor	Size	9,720	9,720
	High	3,360	3,360
	Margin	6,360	6,360
RTI	Size	599,720	599,720
	High	364,680	364,760
	Margin	235,040	234,960
Livingstone	Size	14,720	14,720
	High	10,320	10,320
	Margin	4,400	4,400

In general, the stack memory results agree with the results shown in table 19 (see appendix 8.6). There are minor differences in stack usage for the timer monitor, command monitor, the ROS, and the RTI between the results reported in table 19 and those reported in table 13. For the timer and command monitors, the differences are most likely due to the nature of running these tests in real-time. The minor differences in the ROS stack memory results can be attributed to a simple modification that was implemented in the GPU for the system-level tests but not for the generic model tests. This modification was the addition of a carriage return to one of the output lines in the ROS code. Lastly, the differences in the RTI stack memory results can be attributed to the subsystem interaction table, which is compiled into the code and is different for the two models.

Dynamic memory

The generic model dynamic memory results are shown in table 14, and they agree with the results for the PITEX 2.0 model, which were shown in table 9.

TABLE 14.—GENERIC MODEL DYNAMIC MEMORY RESULTS

Scenario	Livingstone (bytes)	RTI (bytes)	Monitors				
			Command (bytes)	Set point (bytes)	Valve timer (bytes)	Timer (bytes)	TIS (bytes)
Nominal	1048680	0	5144	98248	19616	320	123104
NFS18	1048800	0	5144	98248	19616	440	123104

Timing

The generic model timing result for NFS18 is shown in table 15. The diagnostic delay is 0.08 sec slower than with the PITEX 2.0 model. This is not significant and is equal to one frame update of the data. Therefore, with the generic model, at least for this one scenario, there is no degradation in the diagnostic performance.

TABLE 15.—TIMING RESULTS FOR NFS18 WITH THE GENERIC MODEL

Scenario	To (sec)	Tl (sec)	Diagnostic delay (sec)
NFS18	9735.290	9738.320	3.030

6.0 Concluding Remarks

In general, the capability to do on-board real-time diagnostics has not been demonstrated on problems that have the level of complexity and uncertainty that would represent a real space transportation application. To address this technology gap, PITEX developed and applied health management technologies on a relevant 2GRLV test article, namely the X-34 Main Propulsion System (MPS), with the software diagnostic system processing flight-like data and operating in real time on flight-like hardware. Using X-34 domain experts from NASA Marshall Space Flight Center and Orbital Sciences Corporation (OSC), critical failure modes were identified. These propulsion system failures, such as valves sticking open or closed, regulator problems, and sensor and microswitch failures, were injected at various points in a simulated mission. In addition, the capability to do reliable diagnostics in the presence of expected nominal hardware variations was taken into account by incorporating such expected variations, based on OSC data, for a subsystem of the MPS. In order to accurately replicate flight data, noise was superimposed on the simulation output, micro-switch information was added, and sensor resolution was taken into consideration, all of which provided realistic sensor signals. These simulated data were processed, in real time, by diagnostic software running on a commercial grade version of actual flight hardware. In all cases, PITEX detected and isolated the injected fault correctly without any false alarms. Furthermore, the risks to hardware and safety concerns were addressed by quantifying resource management needs and performance under CPU restrictions. Analysis showed that the PITEX system used fewer resources than were allocated for the health management computation, indicating that the diagnostic system could be expanded to cover additional components.

PITEX has confidently demonstrated the capability to perform real-time propulsion system diagnostics accurately and reliably. When incorporated on board vehicles, this capability will eliminate delays, extra communication bandwidth, and human effort requirements associated with the telemetry of data to the ground for diagnostic analysis. As originally envisioned, PITEX-like technologies were to be applied to the X-34 during the manned captive carry phase of the mission and were expected to make significant enhancements to safety. While safety improvements are difficult to quantify, it is envisioned that by having highly reliable fault detection in real-time, corrective actions can be taken in a timely manner to maintain safe transportation system operation. In addition, the expected operational cost savings and improved fault detection capabilities will allow PITEX technologies to significantly improve the affordability and reliability of reusable space transportation systems.

7.0 References

1. Meyer, C.M., et al.; Propulsion IVHM Technology Experiment Overview, 2003 IEEE Aerospace Conference.
2. Balaban, E., et al.; Transient Region coverage in the Propulsion IVHM Technology Experiment, 2004 International Conference on Artificial Intelligence.
3. Maul, W.A., et al.; Addressing the Real-World Challenges in the Development of Propulsion IVHM Technology Experiment (PITEX), AIAA–2004–6361.
4. Sgarlata, P.K. and Winters, B.A.; X-34 Propulsion System Design, AIAA–97–3304.
5. Champion, R.H. and Darrow, R.J.; X-34 Main Propulsion System Design and Operation, AIAA–98–4032.
6. Brown, T.M., et al.; Propellant Management and Conditioning within the X-34 Main Propulsion System, AIAA–98–3518.
7. McDonald, J.P., et al.; Propellant Feed Subsystem for the X-34 Main Propulsion System, AIAA–98–3517.
8. Hedayat, A., et al.; Pressurization, Pneumatic, and Vent Subsystems of the X-34 Main Propulsion System, AIAA–98–3519.
9. Arnold, R.; Low Cost Boost Technologies Fastrac 60K Engine to X-34 Vehicle Interface Control Document, October 27, 1998.
10. Eastern and Western Range (EWR) 127-1 Safety Requirements, October 31, 1997.

8.0 Appendix

The following appendices provide technical data, information, and test results that support the main body of this report.

8.1 Acronym List

API	Application Program Interface
ARC	Ames Research Center
Bss	Block Storage Space
CBFS	Conflict-Directed Best First Search
CPU	Central Processing Unit
EEPROM	Electronically Erasable Programmable Read Only Memory
IFSn	IVTB Fault Scenario number ‘n’
gcc	GNU’s C compiler
GHe	Gaseous Helium
GPU	Ground Processing Unit
GRC	Glenn Research Center
GUI	Graphical User Interface
I/O	Input/Output
IVHM	Integrated Vehicle Health Management
IVTB	Integrated Vehicle Test Bed
KSC	Kennedy Space Center
LOX	Liquid Oxygen
L2	C++ version of Livingstone
MBR	Model Based Reasoning
MPS	Main Propulsion System
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
NFSn	NITEX Fault Scenario number ‘n’
NGC	Northrop Grumman Corporation
NITEX	NASA IVHM Technology Experiment for X-Vehicles
NVRAM	Non-Volatile Random Access Memory
OSC	Orbital Sciences Corporation
PFSn	PITEX Fault Scenario number ‘n’
PITEX	Propulsion IVHM Technology Experiment
RAM	Random Access Memory
RFU	Real-time Flight Unit
ROCETS	Rocket Engine Transient Simulation
ROS	Results Output System
RP-1	Rocket-Propellant 1 (kerosene-based rocket fuel)
RTI	Real Time Interface
RxU	Real-Time x Unit
SCSI	Small Computer Serial Interface
SLI	Strategic Launch Initiative
TcL	Tool Command Language
TCP	Transmission Control Protocol
TIS	Telemetry Input System
VME	Virtual Memory Environment
V&V	Verification and Validation
2GRLV	Second Generation Reusable Launch Vehicle

8.2 X-34 MPS Schematic Nomenclature

Component	Description
MMSW102X	Open Switch (SV02)
MMSW103X	Open Switch (SV03)
MMSW112X	Closed Switch (SV02)
MMSW113X	Open Switch (SV13)
MMSW203X	Open Switch (PV03)
MMSW205X	Open Switch (SV31)
MMSW213X	Closed Switch (PV03)
MMSW301X	Open Switch (PV02)
MMSW303X	Open Switch (SV36)
MMSW304X	Open Switch (PV07)
MMSW305X	Open Switch (PV08)
MMSW311X	Closed Switch (PV02)
MMSW314X	Closed Switch (PV07)
MMSW315X	Closed Switch (PV08)
MPRE101P	Pressurization System Supply Pressure
MPRE102P	Pneumatic System Supply Tank Pressure
MPRE103P	REG1 Outlet Pressure
MPRE104P	Forward LOX Tank Pressurization Inlet Pressure
MPRE105P	RP-1 Tank Pressurization Inlet Pressure
MPRE107P	REG2 Outlet Pressure
MPRE201P	Aft LOX Tank Ullage Pressure
MPRE202P	Forward LOX Tank Ullage Pressure A
MPRE204P	Aft LOX Tank Outlet Pressure
MPRE205P	LOX Feed Flow Meter Upstream Pressure
MPRE206P	LOX Fill/Drain/Dump Flow Meter Upstream Pressure
MPRE207P	LOX Feed Line Pressure
MPRE208P	LOX Filter Differential Pressure
MPRE211P	LOX Feed Line Flow Meter Downstream Pressure
MPRE212P	Forward LOX Tank Ullage Pressure B
MPRE222P	Forward LOX Tank Ullage Pressure C
MPRE301P	RP-1 Ullage Pressure A
MPRE302P	RP-1 Tank Outlet Pressure
MPRE304P	RP-1 Feed Line Filter Upstream Pressure
MPRE305P	RP-1 Feed Line Filter Downstream Pressure
MPRE311P	RP-1 Ullage Pressure B
MPRE321P	RP-1 Ullage Pressure C
MRTD101T	Lower Port Pressurant Tank Temperature
MRTD102T	Pneumatic System Supply Tank Temperature
MRTD103T	Pressurization System Line Temperature
MRTD104T	Forward LOX Tank Pressurization Inlet Temperature
MRTD105T	RP-1 Tank Pressurization Inlet Temperature
MRTD107T	Lower Starboard Pressurant Tank Temperature
MRTD108T	Upper Port Pressurant Tank Temperature
MRTD109T	Upper Starboard Pressurant Tank Temperature

Component	Description
MRTD201T	LOX Intertank Liquid Line Temperature
MRTD203T	LOX Tank Vent Line Temperature
MRTD204T	Aft LOX Tank Outlet Temperature
MRTD205T	LOX Feed Line Temperature
MRTD206T	LOX Drain/Dump Line Temperature
MRTD207T	LOX Feed Flow Meter Inlet Temperature
MRTD301T	RP-1 Vent Line Temperature
MRTD302T	RP-1 Tank Outlet Temperature
MRTD304T	RP-1 Feed Line to Engine Inlet Temperature
MRTD305T	RP-1 Flow Meter Inlet Temperature
PR02	LOX Vent Line Relief Valve
PR03	RP-1 Vent Line Primary Relief Valve
PR04	RP-1 Vent Line Secondary Relief Valve
PR07	LOX Feed Line Pressure Relief Valve
PV02	RP-1 Feed Pneumatic Valve
PV03	LOX Feed Pneumatic Valve
PV07	RP-1 Tank Feed Pneumatic Valve
PV08	RP-1 Tank Fill/Drain/Dump Pneumatic Valve
RG01	Pressurization Secondary Regulator
RG02	Pneumatic Secondary Regulator
RG11	Pressurization Primary Regulator
RG21	Pneumatic Primary Regulator
SV01	Pressurization System Isolation Valve
SV02	RP-1 Tank Primary Pressurization Valve
SV03	LOX Tank Primary Pressurization Valve
SV08	Pneumatic System Isolation Valve
SV12	RP-1 Tank Secondary Pressurization Valve
SV13	LOX Tank Secondary Pressurization Valve
SV31	LOX Vent/Relief Pneumatic Pilot Valve
SV32	RP-1 Feed Pneumatic Pilot Valve
SV33	LOX Feed Pneumatic Pilot Valve
SV36	RP-1 Vent/Relief Pneumatic Pilot Valve
SV37	RP-1 Tank Feed Pneumatic Pilot Valve
SV38	RP-1 Tank Fill/Drain/Dump Pneumatic Pilot Valve
TK01	Upper Pressurization Tank
TK02	Forward Pneumatic Tank
TK07	Lower Port Pressurization Tank
TK08	Lower Center Pressurization Tank
TK09	Lower Starboard Pressurization Tank
VR01	LOX Tank Vent/Relief Valve
VR06	RP-1 Tank Vent/Relief Valve

8.3 PITEX X-34 MPS Fault Scenarios

During NITEX and the PITEX Base Period, a series of fault scenarios were developed to test the diagnostic system. Descriptions of these scenarios can be found in table 16, and their locations are identified on figure 11. All of the fault scenarios occur during the captive carry mission phase, which was shown in figure 6 (section 3.2.2).

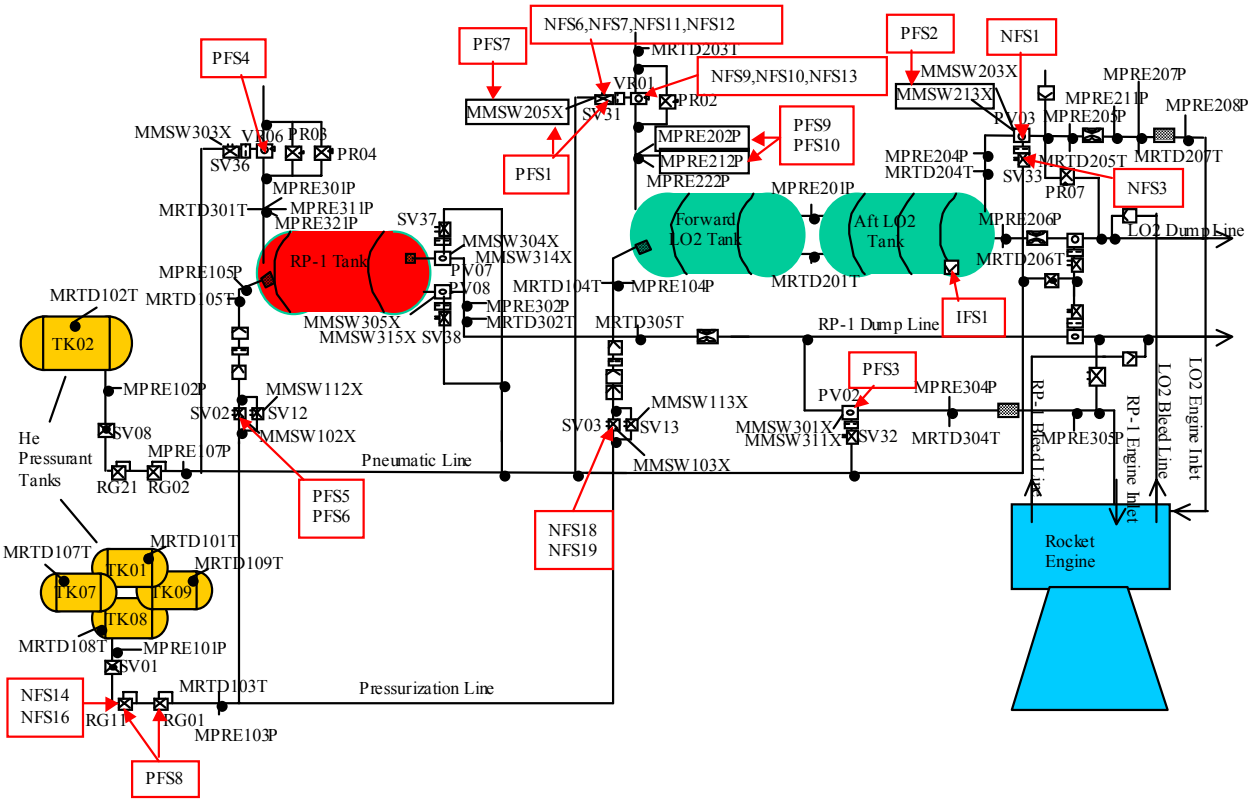

Figure 11.—Locations of the fault scenarios.

Scenarios are designated by an “N” for NITEX, “P” for PITEX, or “I” for IVTB.

TABLE 16.—FAULT SCENARIOS

Scenario	Description	First observation of failure in data set (sec)	End of data set (sec)	Notes
Nominal	Nominal data set	N/A	9776	
NFS1	LOX Feed Valve, PV03, sticks closed.	9410.0	9776	
NFS3	LOX Feed Servo Valve, SV33, fails closed.	9539.79	9618.68	
NFS6	LOX Tank Vent Relief Servo Valve, SV31, sticks open	2706.9	9000	Same results as FS9 with switch/command differences.
NFS7	LOX Tank Vent Relief Servo Valve, SV31, fails open.	5000.0	9000	Same results as FS10 with switch/command differences.
NFS9	LOX Tank Vent Relief Valve, VR01, sticks open.	2706.9	9000	
NFS10	LOX Tank Vent Relief Valve, VR01, fails open.	5000.0	9000	
NFS11	LOX Tank Vent Relief Servo Valve, SV31, sticks closed.	5167.4	9000	Same results as FS13 with switch/command differences.
NFS12	LOX Tank Vent Relief Servo Valve, SV31, fails closed.	3331.0	9000	
NFS13	LOX Tank Vent Relief Valve, VR01, sticks closed.	5167.4	9000	
NFS14	Pressurization Regulator, RG11, fails high.	9000.0	9776	
NFS16	Pressurization Regulator, RG11, fails low.	9642.67	9776	
NFS18	LOX Tank Pressurization Valve, SV03, sticks open.	9735.29	9776	
NFS19	LOX Tank Pressurization Valve, SV03, sticks closed.	9766.05	9776	
PFS1	The open microswitch, MMSW205X, fails on the LOX vent/relief solenoid valve, SV31.	3301.7 and 7260.8	9776	*Intermittent failure.
PFS2	The close microswitch, MMSW213X, fails on the LOX feed valve, PV03.	9410.0	9776	*The microswitches on PV03 contradict each other.
PFS3	The RP-1 feed valve, PV02, fails closed after the RP-1 bleed has been initiated.	9359.0	9409	*PV07 is open. *It is not possible to distinguish between a failure in SV32 and PV02.
PFS4	The RP-1 vent/relief pneumatic valve, VR06, fails open.	9379.0	9409	*No venting of the RP-1 tank should be required during captive carry. *SV01 is open.
PFS5	The primary RP-1 tank pressurization valve, SV02, sticks closed.	9383.86	9409	
PFS6	The primary RP-1 tank pressurization valve, SV02, sticks open.	9384.71	9409	
PFS7	The open microswitch, MMSW205X, fails on the LOX vent/relief solenoid valve, SV31. After that, SV31 fails closed.	3301.7	9000	*Double fault.
PFS8	GHe pressurization system pressure regulators, RG11 and RG01, both regulate high.	9000.0	9770	*Double fault. *Above 370 psia.
PFS9	Two of the LOX vent line pressure sensors, MPRE202P and MPRE212P, fail high.	0.0	9000	Two sensors would need to fail in the same direction to cause this fault.
PFS10	Two of the LOX vent line pressure sensors, MPRE202P and MPRE212P, fail low.	0.0	9000	Two sensors would need to fail in the same direction to cause this fault.
IFS1	The last lower flapper valve in the aft LOX tank fails shut.	9199.0	9776	

8.4 Robustness Tests

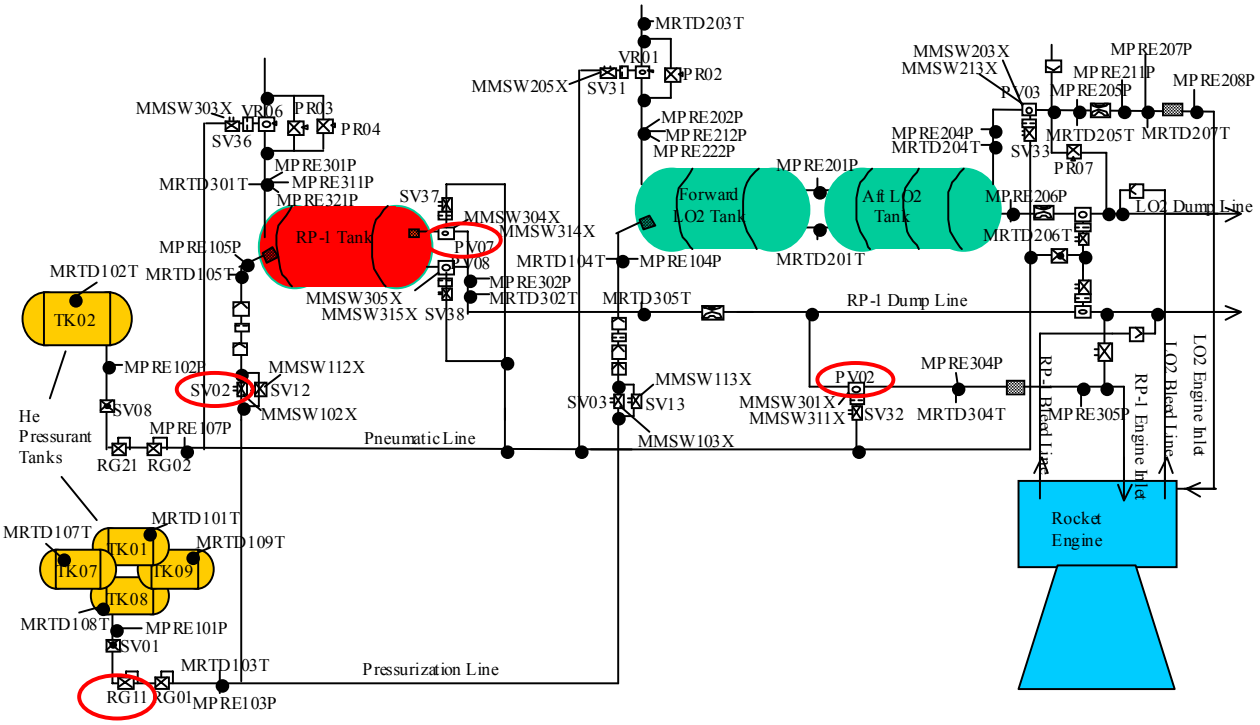

Figure 12.—Location of RP-1 Subsystem Parameters.

TABLE 17.—NOMINAL PARAMETER VARIATION

Test	RG11 setpoint (psia)	PV07 opening (sec)	PV02 opening (sec)	SV02 opening/closing (sec)
1	332.5	0.3	0.3	0.025
2	367.5	0.3	0.3	0.025
3	332.5	3	0.3	0.025
4	367.5	3	0.3	0.025
5	332.5	0.3	3	0.025
6	367.5	0.3	3	0.025
7	332.5	3	3	0.025
8	367.5	3	3	0.025
9	332.5	0.3	0.3	0.05
10	367.5	0.3	0.3	0.05
11	332.5	3	0.3	0.05
12	367.5	3	0.3	0.05
13	332.5	0.3	3	0.05
14	367.5	0.3	3	0.05
15	332.5	3	3	0.05
16	367.5	3	3	0.05
17	350	3	0.3	0.05
18	350	0.3	3	0.05
19	350	3	3	0.05
20	350	0.3	0.3	0.025
21	350	3	0.3	0.025
22	350	0.3	3	0.025
23	350	3	3	0.025

8.5 Validation Results

Table 18 lists the results from the Option 1 system-level validation testing. The table shows the fault injection time for each scenario and the diagnostic time when the set of fault candidates was detected and reported. For each fault scenario, there may be multiple sets of fault candidates reported in table 18. Each fault candidate set represents the current diagnostic analysis based upon the available observations from the system data. The highlighted field identifies the injected fault candidate.

TABLE 18.—VALIDATION RESULTS

Fault scenario	Fault injection time (sec)	Diagnostic time (sec)	Candidates	Rank
NFS1	9410.00	9413.20	SV33 Stuck Closed	2
			PV03 Stuck Closed	2
			SV33 Unknown Fault	5
			PV03 Unknown Fault	5
			PV03 Close MS Faulty PV03 Open MS Faulty	6
		9417.28	SV33 Stuck Closed	2
			PV03 Stuck Closed	2
			SV33 Unknown Fault	5
			PV03 Unknown Fault	5
NFS3	9539.79	9544.40	SV33 Stuck Closed	2
			PV03 Stuck Closed	2
			SV33 Unknown Fault	5
			PV03 Unknown Fault	5
			PV03 Close MS Faulty PV03 Open MS Faulty	6
		9550.0	SV33 Stuck Closed	2
			PV03 Suck Closed	2
			SV33 Unknown Fault	5
			PV03 Unknown Fault	5
NFS6	2706.90	2710.00	SV31 Open MS Faulty	3
			SV31 Stuck Open	3
			SV31 Unknown Fault	5
		2712.88	SV31 Stuck Open	3
			SV31 Unknown Fault	5
			SV31 Open MS Faulty VR01 Stuck Open	5
			SV31 Open MS Faulty MRTD203T Faulty	5
		2714.00	SV31 Stuck Open	3
			SV31 Unknown Fault	5
			SV31 Open MS Faulty VR01 Stuck Open	5
NFS7	5000.00	5010.08	SV31 Stuck Open	3
			SV31 Unknown Fault	5
			SV31 Open MS Faulty VR01 Stuck Open	5
			SV31 Open MS Faulty MRTD203T Faulty	5
		5022.32	SV31 Stuck Open	3
			SV31 Unknown Fault	5
			SV31 Open MS Faulty VR01 Stuck Open	5
NFS9	2706.90	2712.96	VR01 Stuck Open	2
			MRTD203T Faulty	2
			SV31 Open MS Faulty SV31 Stuck Open	6

Fault scenario	Fault injection time (sec)	Diagnostic time (sec)	Candidates	Rank
		2714.00	VR01 Stuck Open	2
			SV31 Open MS Faulty SV31 Stuck Open	6
NFS10	5000.00	5010.80	VR01 Stuck Open	2
			MRTD203T Faulty	2
			SV31 Stuck Open SV31 Open MS Faulty	6
		5022.40	VR01 Stuck Open	2
			SV31 Stuck Open SV31 Open MS Faulty	6
NFS11	5167.40	5170.48	SV31 Stuck Closed	2
			SV31 Open MS Faulty	3
			SV31 Unknown Fault	5
		5174.48	SV31 Stuck Closed	2
			SV31 Open MS Faulty VR01 Stuck Closed	5
			SV31 Unknown Fault	5
NFS12	3331.00	3333.68	SV31 Open MS Faulty	3
			SV31 Stuck Closed VR01 Stuck Open	4
		3344.72	SV31 Stuck Closed	2
			SV31 Open MS Faulty VR01 Stuck Closed	5
			SV31 Unknown Fault	5
NFS13	5167.40	5174.56	VR01 Stuck Closed	1
			MPRE107P Faulty RG02 Reg Low	4
			MPRE107P Faulty RG21 Reg Low	4
			SV31 Open MS Faulty SV31 Stuck Closed	5
NFS14	9000.00	9010.32	MPRE103P Faulty	2
			RG11 Reg High	2
			RG11 Faulty	4
			RG01 Faulty	4
NFS16	9642.67	9645.92	MPRE103P Faulty	2
			RG11 Reg Low	2
			RG01 Reg Low	2
			RG11 Faulty	4
			RG01 Faulty	4
NFS18	9735.29	9738.24	SV03 Open MS Faulty	3
			SV03 Stuck Open	3
			SV03 Unknown Fault	5
NFS19	9766.05	9769.20	SV03 Stuck Closed	2
			SV03 Open MS Fault	3
			SV03 Unknown Fault	5
PFS1	3301.70	3304.80	SV31 Stuck Closed	2
			SV31 Open MS Faulty	3
			SV31 Unknown Fault	5
		3320.16	SV31 Open MS Faulty	3
			VR01 Stuck Open SV31 Stuck Closed	4
		3592.88	SV31 Open MS Faulty	3
			VR01 Stuck Open SV31 Stuck Closed VR01 Stuck Closed	5

Fault scenario	Fault injection time (sec)	Diagnostic time (sec)	Candidates	Rank
		4209.60	SV31 Open MS Faulty	3
PFS2	9410.00	9413.44	PV03 Close MS Faulty	3
			SV33 Stuck Closed PV03 Open MS Faulty	5
			PV03 Stuck Closed PV03 Open MS Faulty	5
		9435.36	PV03 Close MS Faulty	3
PFS3	9359.00	9369.28	SV32 Stuck Closed	2
			PV02 Stuck Closed	2
			SV32 Unknown Fault	5
			PV02 Unknown Fault	5
			RG21 Reg Low PV07 Stuck Open MPRE107P Faulty	6
			RG02 Reg Low PV07 Stuck Open MPRE107P Faulty	6
		9382.08	SV32 Stuck Closed	2
			PV02 Stuck Closed	2
			SV32 Unknown Fault	5
			PV02 Unknown Fault	5
			PV02 Close MS Faulty PV02 Open MS Faulty	6
			RG02 Reg Low PV07 Stuck Open MPRE107P Faulty	6
PFS4	9379.00	9406.08	VR06 Stuck Open	2
			MPRE103P Faulty	2
			SV02 Stuck Closed SV02 Open MS Fauly	5
			VR06 Unknown Fault	5
PFS5	9383.86	9386.96	SV02 Stuck Closed	2
			SV02 Open MS Faulty	3
			SV02 Unknown Fault	5
PFS6	9384.71	9387.04	SV02 Stuck Open	3
			SV02 Open MS Faulty	3
			SV02 Unknown Fault	5
PFS7	3301.70	3304.80	SV31 Stuck Closed	2
			SV31 Open MS Faulty	3
			SV31 Unknown Fault	5
		3320.16	SV31 Open MS Faulty	3
			VR01 Stuck Open SV31 Stuck Closed	4
		3344.72	SV31 Stuck Closed	2
			SV31 Open MS Faulty VR01 Stuck Closed	4
			SV31 Unknown Fault	5
PFS8	9000.00	9010.32	MPRE103P Faulty	2
			RG11 Reg High RG01 Reg High	4
			RG01 Faulty	5
			RG01 Reg High RG11 Faulty	6
PFS9	0.00	10.0	MPRE222P Biased	2
			MPRE222P Faulty	2

Fault scenario	Fault injection time (sec)	Diagnostic time (sec)	Candidates	Rank
			MPRE212P Biased MPRE202P Biased	4
			MPRE212P Biased MPRE202P Faulty	4
			MPRE212P Faulty MPRE202P Biased	4
			MPRE212P Faulty MPRE202P Faulty	4
		5016.56	MPRE212P Biased MPRE202P Biased	4
			MPRE222P Biased MRTD201T Faulty	4
			MPRE212P Biased MPRE202P Faulty	4
			MPRE212P Faulty MPRE202P Biased	4
			MPRE212P Faulty MPRE202P Faulty	4
			MPRE222P Faulty MRTD201T Fault	4
			MPRE222P Faulty MPRE212P Biased MPRE202P Biased	6
PFS10	0.00	10.0	MPRE222P Biased	2
			MPRE222P Faulty	2
			MPRE212P Biased MPRE202P Biased	4
			MPRE212P Biased MPRE202P Faulty	4
			MPRE212P Faulty MPRE202P Biased	4
			MPRE212P Faulty MPRE202P Faulty	4
		2215.28	MPRE212P Biased MPRE202P Biased	4
			MPRE222P Biased MRTD201T Faulty	4
			MPRE212P Biased MPRE202P Faulty	4
			MPRE212P Faulty MPRE202P Biased	4
			MPRE212P Faulty MPRE202P Faulty	4
			MPRE222P Faulty MPRE212P Biased MPRE202P Biased	6
		2246.96	MPRE222P Biased	2
			MPRE222P Faulty	2
			MPRE212P Biased MPRE202P Biased	4
			MPRE212P Biased MPRE202P Faulty	4
			MPRE212P Faulty MPRE202P Biased	4
			MPRE212P Faulty MPRE202P Faulty	4

Fault scenario	Fault injection time (sec)	Diagnostic time (sec)	Candidates	Rank
		2257.28	MPRE212P Biased MPRE202P Biased	4
			MPRE222P Biased MRTD201T Faulty	4
			MPRE212P Biased MPRE202P Faulty	4
			MPRE212P Faulty MPRE202P Biased	4
			MPRE212P Faulty MPRE202P Faulty	4
			MPRE222P Faulty MPRE212P Biased MPRE202P Biased	6
IFS1	9199.00	9235.38	MPRE204P Faulty	2
			Aft LO2 Tank Unknown Fault	6
		9417.2	MPRE204P Faulty MPRE205P Faulty	4
			Aft LO2 Tank Unknown Fault	6
		9419.92	MPRE204P Faulty	2
			Aft LO2 Tank Unknown Fault	6
		9427.6	MPRE204P Faulty MPRE205P Faulty	4
			Aft LO2 Tank Unknown Fault	6
	Identifies injected fault			

8.6 Stack Memory Results

TABLE 19.—STACK MEMORY RESULTS

Tasks		Stack Memory (bytes)													
		Nominal	NFS1	NFS3	NFS6	NFS7	NFS9	NFS10	NFS11	NFS12	NFS13	NFS14	NFS16	NFS18	NFS19
TIS	Size	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728
	High	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368
	Margin	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360
LogTask	Size	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728
	High	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928
	Margin	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800
ROS	Size	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728
	High	5,864	5,776	5,832	5,776	5,880	5,864	5,776	5,864	5,880	5,864	5,776	5,864	5,864	5,776
	Margin	13,864	13,952	13,896	13,952	13,848	13,864	13,952	13,864	13,848	13,864	13,952	13,864	13,864	13,952
TimerMonitor	Size	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720
	High	2,872	2,816	2,816	2,816	2,816	2,816	2,816	2,872	2,816	2,816	2,816	2,816	2,816	2,816
	Margin	6,848	6,904	6,904	6,904	6,904	6,904	6,904	6,848	6,904	6,904	6,904	6,904	6,904	6,904
CmdMonitor	Size	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720
	High	2,728	2,832	2,832	2,728	2,728	2,832	2,728	2,728	2,832	2,832	2,728	2,832	2,728	2,832
	Margin	6,992	6,888	6,888	6,992	6,992	6,888	6,992	6,992	6,888	6,888	6,992	6,888	6,992	6,888
SetPointMonitor	Size	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720
	High	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904
	Margin	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816
vlvTimerMonitor	Size	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720
	High	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360
	Margin	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360
RTI	Size	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720
	High	362,432	362,512	362,488	362,432	362,432	362,432	362,432	362,432	362,432	362,432	362,512	362,432	362,432	362,432
	Margin	237,288	237,208	237,232	237,288	237,288	237,288	237,288	237,288	237,288	237,288	237,208	237,288	237,288	237,288
Livingstone	Size	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720
	High	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320
	Margin	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400

TABLE 19.—STACK MEMORY RESULTS (CONCLUDED)

Task		Stack Memory (bytes)										
		PFS1	PFS2	PFS3	PFS4	PFS5	PFS6	PFS7	PFS8	PFS9	PFS10	IFS1
TIS	**Size**	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728
	High	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368	2,368
	Margin	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360	7,360
LogTask	**Size**	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728	9,728
	High	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928	2,928
	Margin	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800
ROS	**Size**	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728	19,728
	High	5,776	5,864	5,776	5,776	5,880	5,776	5,776	5,832	5,864	5,880	5,864
	Margin	13,952	13,864	13,952	13,952	13,848	13,952	13,952	13,896	13,864	13,848	13,864
TimerMonitor	**Size**	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720
	High	2,872	2,816	2,816	2,816	2,816	2,816	2,816	2,816	2,816	2,816	2,816
	Margin	6,848	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904
CmdMonitor	**Size**	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720
	High	2,728	2,832	2,728	2,832	2,728	2,832	2,832	2,832	2,832	2,728	2,728
	Margin	6,992	6,888	6,992	6,888	6,992	6,888	6,888	6,888	6,888	6,992	6,992
SetPointMonitor	**Size**	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720
	High	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904	6,904
	Margin	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816	7,816
vlvTimerMonitor	**Size**	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720	9,720
	High	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360	3,360
	Margin	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360	6,360
RTI	**Size**	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720	599,720
	High	362,432	362,432	362,432	362,432	362,432	362,432	362,432	362,512	362,488	362,432	362,488
	Margin	237,288	237,288	237,288	237,288	237,288	237,288	237,288	237,208	237,232	237,288	237,232
Livingstone	**Size**	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720	14,720
	High	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320	10,320
	Margin	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400	4,400

8.7 Timing Results

TABLE 20.—HARDWARE TIMING RESULTS

Scenario	Diagnostic delay (sec)
NFS1	60.320
NFS3	15.410
NFS6	20.140
NFS7	15.520
NFS9	20.700
NFS10	11.360
NFS11	20.200
NFS12	15.560
NFS13	20.280
NFS14	10.480
NFS16	3.260
NFS18	26.710
NFS19	20.110
PFS1	20.460
PFS2	20.640
PFS3	12.760
PFS4	21.080
PFS5	20.140
PFS6	24.010
PFS7	20.460
PFS8	10.400
PFS9	15.120
PFS10	15.120

8.8 GPU GUI Displays

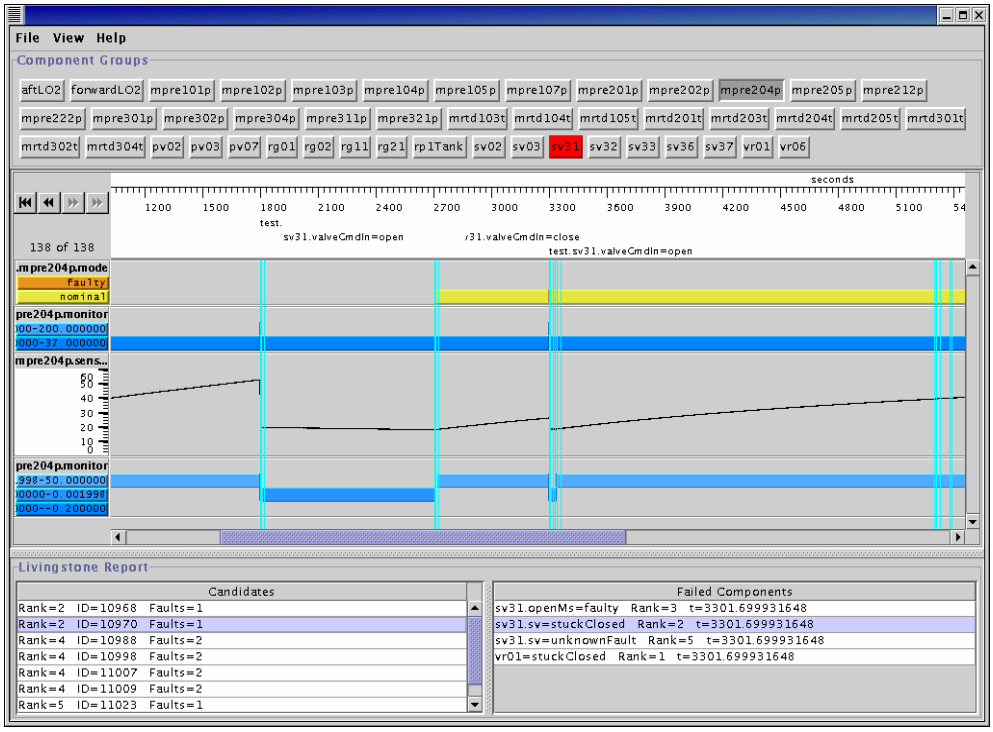

Figure 13.—PFS7: Diagnostic output.

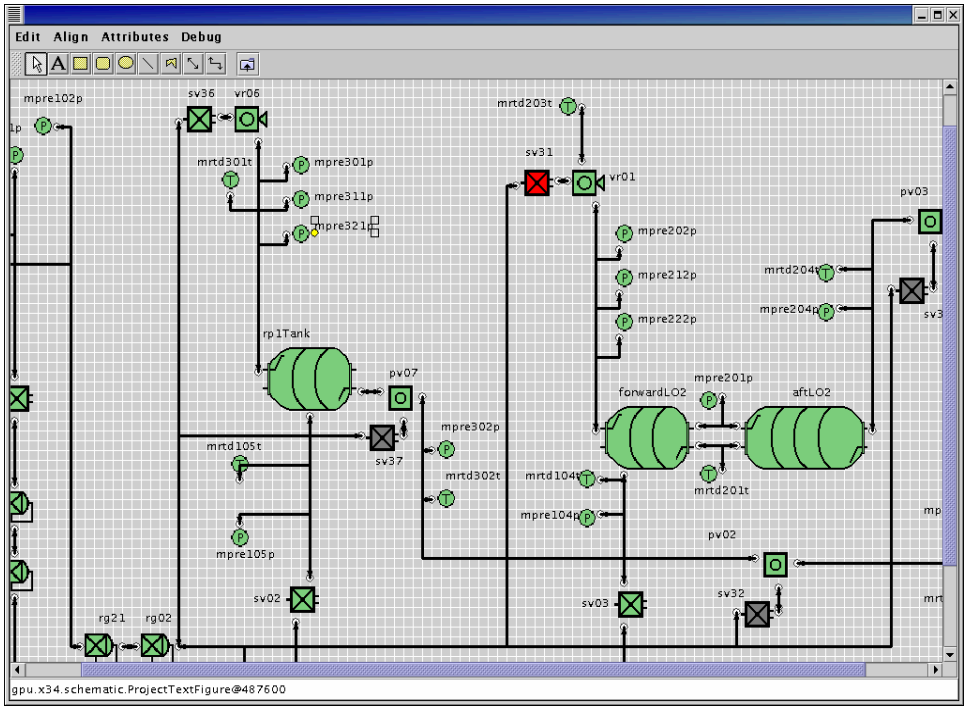

Figure 14.—PFS7: Schematic view of failed component.

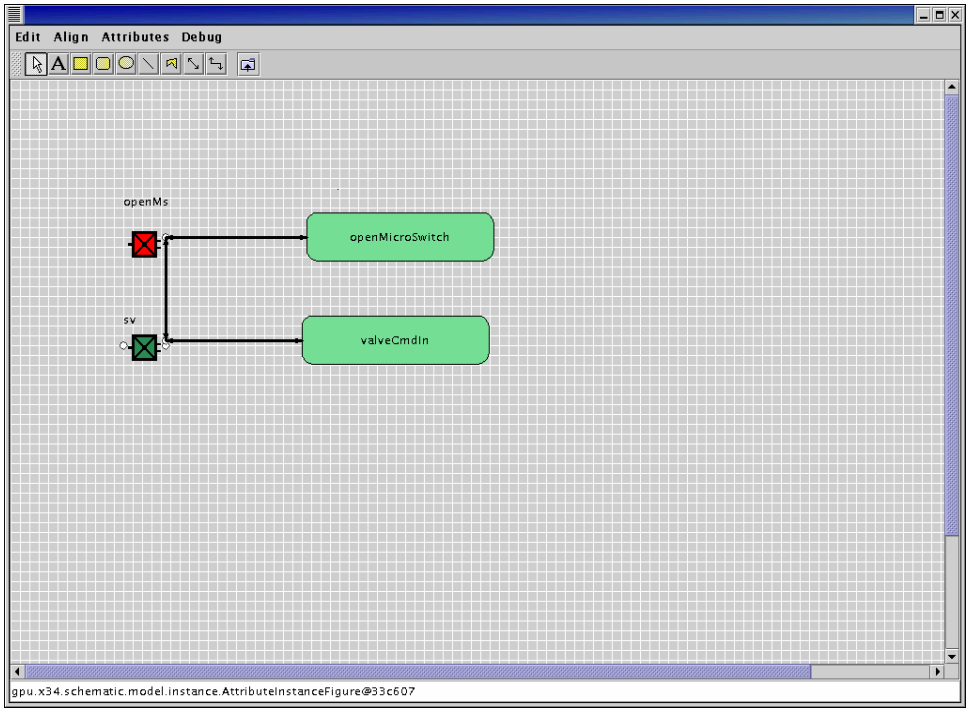

Figure 15.—PFS7: Inside view of failed component.

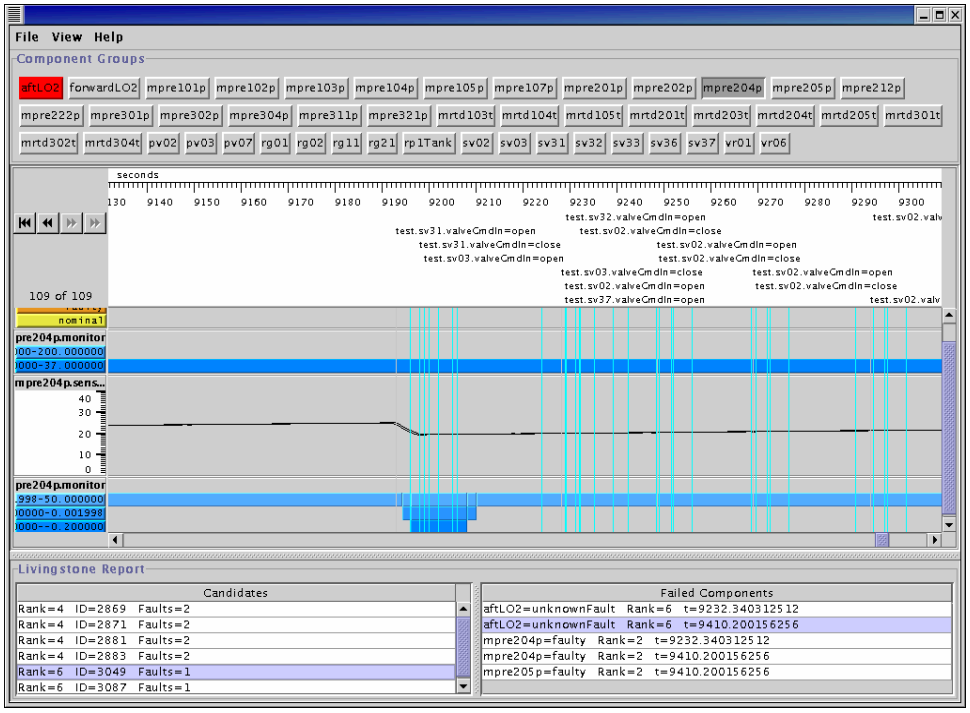

Figure 16.—IFS1: Diagnostic output.

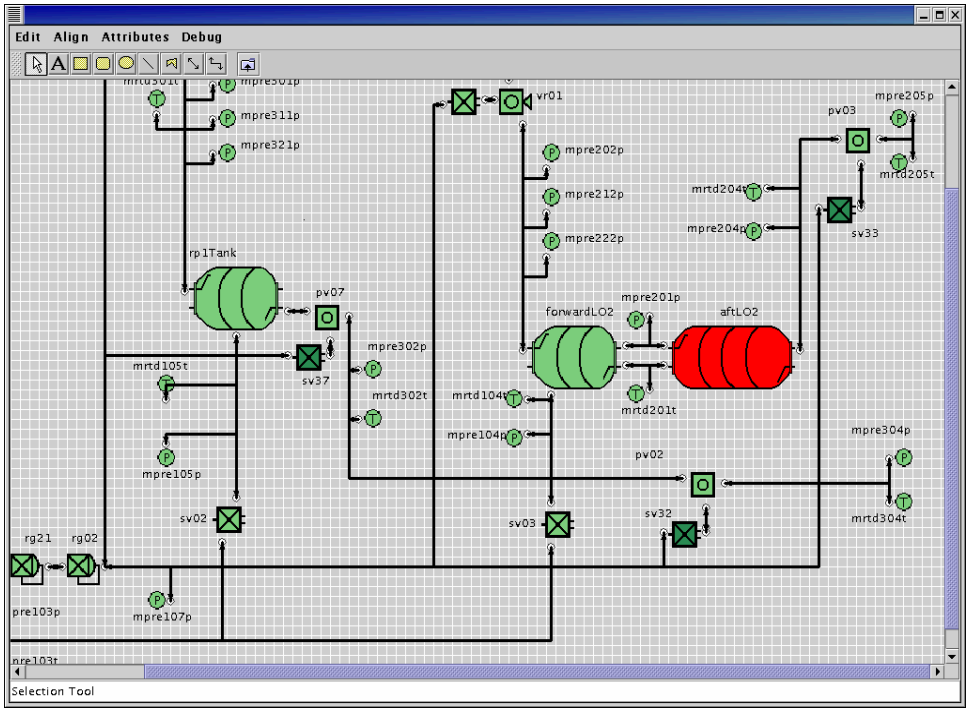

Figure 17.—IFS1: Schematic view of failed component.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 2006	Final Contractor Report

4. TITLE AND SUBTITLE

Propulsion IVHM Technology Experiment

5. FUNDING NUMBERS

WBS–22–794–40–56
NAS3–00145

6. AUTHOR(S)

Amy K. Chicatelli, William A. Maul, and Christopher E. Fulton

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Analex Corporation
1100 Apollo Drive
Brook Park, Ohio 44142

8. PERFORMING ORGANIZATION REPORT NUMBER

E–15484

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2006-214238

11. SUPPLEMENTARY NOTES

Project manager, Edmond Wong, Communications Division, Glenn Research Center, organization code RIC, 216–433–8917.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 20, 60, 65, and 66

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The Propulsion IVHM Technology Experiment (PITEX) successfully demonstrated real-time fault detection and isolation of a virtual reusable launch vehicle (RLV) main propulsion system (MPS). Specifically, the PITEX research project developed and applied a model-based diagnostic system for the MPS of the X-34 RLV, a space-launch technology demonstrator. The demonstration was simulation-based using detailed models of the propulsion subsystem to generate nominal and failure scenarios during captive carry, which is the most safety-critical portion of the X-34 flight. Since no system-level testing of the X-34 Main Propulsion System (MPS) was performed, these simulated data were used to verify and validate the software system. Advanced diagnostic and signal processing algorithms were developed and tested in real time on flight-like hardware. In an attempt to expose potential performance problems, the PITEX diagnostic system was subjected to numerous realistic effects in the simulated data including noise, sensor resolution, command/valve talkback information, and nominal build variations. In all cases, the PITEX system performed as required. The research demonstrated potential benefits of model-based diagnostics, defined performance metrics required to evaluate the diagnostic system, and studied the impact of real-world challenges encountered when monitoring propulsion subsystems.

14. SUBJECT TERMS

System failures; Failure analysis; Fault detection; Systems health monitoring; Aerospace safety; Safety

15. NUMBER OF PAGES

55

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	